REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	9 Dec 96	

4. TITLE AND SUBTITLE
Generalized Predictive Control in the Presence of Constraints

5. FUNDING NUMBERS

6. AUTHOR(S)
Jesse Ross Gossner

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Oxford University

8. PERFORMING ORGANIZATION REPORT NUMBER
96-082

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
DEPARTMENT OF THE AIR FORCE
AFIT/CI
2950 P STEET, BLDG 125
WRIGHT-PATTERSON AFB OH 45433-7765

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

14. SUBJECT TERMS

15. NUMBER OF PAGES
183

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

This thesis, titled "Generalized Predictive Control in the Presence of Constraints," is submitted to the Department of Engineering Science, Oxford University, in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

Jesse Ross Gossner
Merton College
Trinity, 1996

Abstract

Several predictive control strategies which handle physical system constraints have been proposed in the literature. The difficulty with these approaches is that, in an attempt to optimize output tracking over a finite horizon *and* guarantee stability, they define an optimization problem which can be infeasible; in the case of systems with poles and/or zeros outside the unit circle, this can lead to instability. In this thesis we develop stability conditions which ensure the continuing existence of a stabilizing solution and propose improved algorithms which overcome finite horizon infeasibility and give stability and asymptotic tracking.

Previous strategies require an assumption concerning the feasibility of making the output reach its set-point within a finite horizon. Here, we propose two algorithms which overcome these difficulties; the first does so by mixing two- and infinity-norm objectives and the second retains only a two-norm objective, but introduces an additional constraint to ensure stability.

We then examine the necessity of the strategy of requiring the predicted errors and input increments to reach zero within some finite horizon and show that, for a guarantee of stability, one needs only require these predictions to be stable. We propose algorithms which implement these relaxed restrictions and thus yield optimizations which, for a given number of degrees of freedom, utilize the entire class of stabilizing solutions rather than the sub-class of finite length sequences.

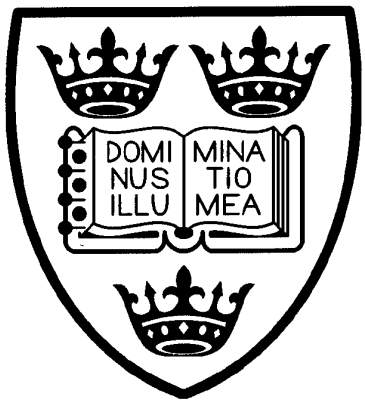
Previous work does not consider the case of systems subject to physical constraints *and* disturbances. To guarantee feasibility, any input/output predictions must take into account the effects of disturbances. The final purpose of this thesis is to derive stability conditions for systems subject to disturbances, and then to develop algorithms which reserve enough control authority to reject the effects of bounded disturbances and thus retain the guarantee of stability.

Generalized Predictive Control in the Presence of Constraints

by

Jesse Ross Gossner

Merton College



Department of Engineering Science
Oxford University

5 June, 1996

Gloria in excelsis Deo

This thesis, titled "Generalized Predictive Control in the Presence of Constraints," is submitted to the Department of Engineering Science, Oxford University, in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

Jesse Ross Gossner
Merton College
Trinity, 1996

Abstract

Several predictive control strategies which handle physical system constraints have been proposed in the literature. The difficulty with these approaches is that, in an attempt to optimize output tracking over a finite horizon *and* guarantee stability, they define an optimization problem which can be infeasible; in the case of systems with poles and/or zeros outside the unit circle, this can lead to instability. In this thesis we develop stability conditions which ensure the continuing existence of a stabilizing solution and propose improved algorithms which overcome finite horizon infeasibility and give stability and asymptotic tracking.

Previous strategies require an assumption concerning the feasibility of making the output reach its set-point within a finite horizon. Here, we propose two algorithms which overcome these difficulties; the first does so by mixing two- and infinity-norm objectives and the second retains only a two-norm objective, but introduces an additional constraint to ensure stability.

We then examine the necessity of the strategy of requiring the predicted errors and input increments to reach zero within some finite horizon and show that, for a guarantee of stability, one needs only require these predictions to be stable. We propose algorithms which implement these relaxed restrictions and thus yield optimizations which, for a given number of degrees of freedom, utilize the entire class of stabilizing solutions rather than the sub-class of finite length sequences.

Previous work does not consider the case of systems subject to physical constraints *and* disturbances. To guarantee feasibility, any input/output predictions must take into account the effects of disturbances. The final purpose of this thesis is to derive stability conditions for systems subject to disturbances, and then to develop algorithms which reserve enough control authority to reject the effects of bounded disturbances and thus retain the guarantee of stability.

Acknowledgements

The author is grateful for the guidance, instruction, and support of his supervisor, Dr. B. Kouvaritakis, and their collaborator, Dr. J.A. Rossiter. The author also wishes to thank Merton College for administrative support, the U.S. Air Force Institute of Technology for financial and administrative support and the U.S. Air Force Academy, Department of Astronautics for sponsorship. Most importantly, the author offers his thanks and love to his wife, Susan; without her understanding, support, and assistance, this thesis would not exist.

Table of Contents

List of Figures

List of Symbols

Acronym	Description
GPC	Generalized Predictive Control
SGPC	Stable Generalized Predictive Control
CSGPC	Constrained Stable Generalized Predictive Control
MWLS	Mixed Weights Least Squares
MCSGPC	Modified Constrained Stable Generalized Predictive Control
CaSC	Cautious Stable Control
CaML	Cautious Mean Level Stable Control
IHSPC	Infinite Horizon Stable Predictive Control
CCaSC	Constrained Cautious Stable Control
LP/QP	Linear/quadratic program(ming)
FLS/ILS	Finite/infinite length sequence
STF/STIF	Short term feasibility/infeasibility
LTF/LTIF	Long term feasibility/infeasibility
LHS/RHS	Left/right hand side
DOF	Degree(s) of freedom

Symbol	Description
$a(z)$	Open-loop system denominator polynomial
n_a	Order of $a(z)$
$b(z)$	Open-loop system numerator polynomial
n_b	Order of $b(z)$
n_y	Output horizon
$y(z)/\underset{\rightarrow}{y}$	Polynomial/vector whose coefficients/elements are the *future* values of the output: y_{t+i}, $0<i\leq n_y$
$\underset{\leftarrow}{y}$	Vector whose elements are the past values of the output: y_{t-i}, $0\leq i\leq n_a$
n_u	Input horizon
$\Delta u(z)/\underset{\rightarrow}{\Delta u}$	Polynomial/vector whose coefficients/elements are the current and *future* values of the input increment: Δu_{t+i}, $0\leq i<n_u$
$\underset{\leftarrow}{\Delta u}$	Vector whose elements are the past values of the output: Δu_{t-i}, $0<i\leq n_b$
n_c	Command horizon
$c(z)/\underset{\rightarrow}{c}$	Polynomial/vector whose coefficients/elements are *future* values of the command input (DOF available for optimization)
c_∞	Steady-state value of command input (DOF) polynomial, c_{t+i}, for $i\geq n_c$
n_r	Reference horizon
n_1	Initial output horizon
$\mathbf{1}$	Vector of ones: $\mathbf{1}=[1\ldots1]^T$
$\mathbf{0}$	Vector of zeros: $\mathbf{0}=[0\ldots0]^T$
e_i	Vector with the i^{th} element equal to 1 and all other elements zero: eg. $e_1=[1\ 0\ldots0]^T$

Symbol	Description
C_p	Toeplitz convolution matrix of $p(z)$: $C_p = \begin{bmatrix} p_0 & 0 & \cdots & 0 \\ \vdots & p_0 & 0 & \vdots \\ p_{n_n} & \vdots & p_0 & 0 \\ 0 & p_{n_n} & \cdots & p_0 \end{bmatrix}$
Γ_p	First n columns of C_p
Θ_p	Remaining columns of C_p: $C_p = [\Gamma_p \quad \Theta_p]$
θ_p	Column sum of Θ_p: $\theta_p = \Theta_p \mathbf{1}$
H_p	Hankel matrix of $p(z)$: $H_p = \begin{bmatrix} p_1 & \cdots & p_{n_n} & 0 \\ \vdots & & 0 & \\ p_{n_n} & 0 & & \vdots \\ 0 & & \cdots & 0 \end{bmatrix}$
U_o	Centre of input absolute constraint limit
U	Radius of input absolute constraint limit
R	Input rate constraint limit
A	"Free" part of normalized constraint inequality
$v(t)$	Fixed part of normalized constraint inequality
F_{n_c}	Constraint feasibility region with n_c dimensions

Chapter 1

Introduction

Model Based Predictive Control (see reviews of [12], [26]) owes its popularity to its simple strategy (the minimization of the predicted tracking errors and control activity) and the fact that it can handle system constraints (eg. [46], [52]). Early work ([31], [9], [29], [6], [7]) lacked a general stability theory, but more recent algorithms ([18], [8], [19], [27], [44]) provide the missing guarantees of stability and can be extended to handle system constraints ([33], [53], [23]).

The basis of predictive control is to predict future tracking errors through some output horizon, n_y, as a function of n_u non-zero future input increments (and past data) and then to minimize a cost involving the two-norm of these predicted errors and input increments. The first optimum input increment is implemented, and then the entire procedure is repeated at the next time instant with new plant data; this procedure is also termed receding horizon control. Unfortunately, as originally formulated, this technique can only be guaranteed to give stable results for special cases. To remedy this, several strategies which adopt the basic idea of Generalized Predictive Control (GPC) [7] have been proposed. Constrained Receding Horizon Predictive Control (CRHPC) [8] adds further terminal constraints, as does a similar algorithm proposed in [27]. Stable

Generalized Predictive Control (SGPC) [19], on the other hand, forms a stabilizing loop around the system before applying GPC to a closed loop configuration which gives finite length sequences (FLSs). By using FLSs, the minimization of the error norm gives a monotonically decreasing cost which guarantees stability and asymptotic tracking.

These properties carry over to the case of predictive control with constraints so long as the implied optimization problem (a quadratic programming problem in Quadratic Programming Generalized Predictive Control, QPGPC, see Ref. [46], or a mixed weights least squares, MWLS, problem in Constrained Stable Generalized Predictive Control, CSGPC, see Ref. [33]) is feasible. This is a strong assumption because it requires "short term" feasibility (STF) which means feasibility over the finite horizons. The fact that a system output can eventually reach a target value without violating any constraints (long term feasibility, or LTF) does not imply it can do so in n_y steps and with only n_u non-zero input increments (STF). What is worse, in some cases, both QPGPC and CSGPC, like other constrained predictive control algorithms, can, in fact, destabilize what was originally a LTF problem; by requiring the predicted output to reach its target within n_y steps, it may be necessary to drive the controls and/or their increments at their limits. If, however, a system is unstable and/or non-minimum phase, these earlier control moves may require future stabilizing control moves which will not be available within the existing constraints; this will lead to instability.

1.1 Thesis overview

Chapter 2 presents a brief review of SGPC and CSGPC, the framework under which this research is carried out.

The problem of feasibility is investigated in Chapter 3, and necessary and sufficient conditions for stability are developed. These conditions are at first *a posteriori* conditions in that they are based on past data; as such, they are only useful for analysis and do not lend themselves to the purposes of control. We, therefore, consider the problem of propagating the stability conditions forward in time and thus propose a procedure for the derivation of *a priori* stability conditions which, for systems with only one unstable pole, provide explicit conditions for retaining LTF. Obviously, *any* control strategy which ignores these conditions and violates them will be unstable. Later, we will demonstrate the use of these explicit *a priori* conditions in the presence of disturbances. Chapter 3 then goes on to develop general stability conditions for systems with any number of unstable poles; the result relies on the use of linear programming, but, for a given number of degrees of freedom, provides conditions which are both necessary and sufficient for stability.

The results of Chapter 3 provide conditions on the current input which guarantee stability, but they do not, in themselves, lead to an algorithm which yields an optimum choice for that input. In Chapter 4, we consider an alternative procedure for dealing with short term infeasibility (STIF) by focusing on CSGPC and proposing modifications which maintain stability when, due to set-point changes, CSGPC encounters STIF. The problem of STIF has been addressed previously ([2], [14]) by fixing the closed-loop controller and using non-linear prefiltering to condition the set-point change such that STF is retained. This approach is computationally simple, but optimizes tracking of the conditioned, rather than actual, set-point and hence is suboptimal.

CSGPC normally has guaranteed stability and asymptotic tracking; however, no such guarantees can be given in the case of STIF. Because this infeasibility is caused by

the requirement that the output should reach its target value within n_y steps, this requirement must be relaxed. However, the stability proof of SGPC and CSGPC depends on the property implied by the use of FLSs that the output settles after n_y steps. Thus, an obvious strategy to follow is to: (i) retain this last property, but (ii) allow the value to which the output settles at to become a degree of freedom. This is first implemented by changing the objective appropriately so as to minimize the deviation of the steady-state value of the output from its target value. The modified algorithm is activated only when CSGPC is STIF, and guarantees recovery of STF; this, together with the properties of CSGPC, guarantee stability and asymptotic tracking. While this approach has the advantage of simplicity, the modified algorithm does not place any penalty on the norm of the vector of predicted errors during the feasibility recovery stage, and this may degrade transient performance. We, therefore, propose two additional modifications to CSGPC which share in common with the previous modification the property that they are guaranteed to recover STF, but also retain in the objective a component which penalizes deviation of the predicted output values from *actual* set-point values.

Recent work by Zheng and Morari [53] and Allwright [1] minimize the infinity-norm of the predicted errors rather than the two-norm, as CSGPC does. Stability is guaranteed without recourse to terminal constraints, and therefore without the need for a STF assumption, however, the results are restricted to open-loop stable systems. Within the context of SGPC, this work can be extended to unstable systems. The performance of such systems, though, is often not as good as CSGPC, as all effort is spent minimizing just the largest error, and this is often the first one. The system is driven very hard and the responses can be oscillatory. We, therefore, propose a procedure for dealing with STIF in which the objective is usually the standard CSGPC

two-norm minimization, but when STIF is encountered, the set-point is allowed to become a degree of freedom, just as in the other modifications, and the objective is shifted to the minimization of the infinity-norm of the predicted errors until STF is regained. Thus, by mixing objectives, the superior performance of CSGPC is retained when possible, but stability and asymptotic tracking are guaranteed with only the assumption of LTF. The final modification is similar, but retains a two-norm cost when CSGPC is STIF and guarantees an asymptotic return to STF by requiring that the deviation of the predicted steady-state value of the output from the set-point get smaller at each subsequent time step.

In Chapter 5, we look specifically at the necessity of the terminal constraints. Forcing the predicted trajectories of both output errors and control increments to be finite length sequences (FLS) provides a convenient way to guarantee the stability of predictive control strategies, but it has been recognised that for the purposes of stability one actually only needs to force the output errors to be stable ([30], [54]), thereby turning the predicted output error trajectory into an infinite length sequence (ILS). Here, we show that it is also the case that one only needs to force the input increments to be stable, with the effect of getting a predicted input increment trajectory which is ILS; we propose algorithms which implement these changes in philosophy and derive two procedures for calculating an infinite horizon cost involving the sum of the square of the ILS errors and input increments.

Of course, for systems subject to physical constraints, ILS trajectories lead to a practical difficulty; the physical constraints must be invoked over an infinite horizon. This problem is overcome through the use of suitable input/output horizon bounds. A set of such bounds with respect to output constraints have been proposed elsewhere ([30],

[54]). Here, we are concerned with input constraints only, but we explore the use of ILS predictions for both inputs and outputs; therefore, we require bounding results on inputs rather than outputs. We develop simple input bounding techniques which provide an efficient means of invoking the constraints over a infinite horizon by enforcing them over a finite horizon.

Earlier work considers the disturbance-free case. The final results of this thesis, presented in Chapter 6, deal with the inherent clash between disturbances and constraints. We first consider how the effects of disturbances can be propagated forward in time. We then show that the stability results of Chapter 3 carry over to the more general case of systems which are subject to disturbances. Under some assumption of norm-boundedness, we derive the necessary and sufficient *a posteriori* stability conditions in the presence of disturbances. As in Chapter 3, we then push these conditions one step into the future and show how they can be used to avoid instability.

While this gives explicit stability conditions, it is restricted to systems with at most one unstable pole, and does not lead to suitable algorithms, because it applies to infinite horizons and only gives a bound on the current input. We, therefore, develop necessary and sufficient limits on the size of the future inputs required to reject all possible (norm-bounded) future disturbances and then modify the constraint limits of CSGPC so as to reserve this necessary control authority and thus derive an algorithm with guaranteed stability and asymptotic tracking for systems subject to disturbances.

The application of all results are illustrated by numerical examples. Chapter 7 summarizes these results and then discusses some of the open problems in the areas addressed by this thesis.

Chapter 2

The CSGPC Framework

In this chapter, the framework which forms the starting point of this research is briefly reviewed. Section 2.1 introduces Stable Generalized Predictive Control (SGPC), a guaranteed stable method of predicting and minimizing the two-norm of the future errors and input increments up to a receding horizon, for systems not subject to input constraints. SGPC is conceptualized by first placing a stabilizing loop around the plant so that the transfer functions from commanding input c to system input u and from commanding input to system output y are z-transforms of finite length sequences (FLSs). This provides a very simple formulation for the FLS (and therefore stable) output/input predictions in terms of the available degrees of freedom (DOF) which, in fact, are the future values of c. Armed with these predictions, the implementation of SGPC is a simple matter of minimising, with respect to the DOF, a two-norm cost of the predicted future errors and input increments, implementing the first optimum input increment, and then repeating the optimization at the next time step with new plant data. It is well known that LQ control gives stable results when it employs an infinite horizon. This is because the performance index can be shown, in the absence of set-point changes, to be a stable Lyapunov function. SGPC effectively deploys the same stability mechanism, and

thus a key element in the design strategy is the use of an infinite costing horizon. This is easily done with the FLS predictions described above. When the system inputs are unconstrained, SGPC can be expressed as a fixed term controller; Section 2.2 derives this optimal controller and shows that degrees of freedom are available which can be used to improve properties such as stability robustness and noise handling without affecting performance. Section 2.3 then adds input constraints, detailing how these constraints are written in terms of the available DOF and providing a mixed weight least squares (MWLS) iteration to minimize the future predicted errors and input increments subject to the input constraints. The resulting algorithm, Constrained Stable Generalized Predictive Control (CSGPC), is otherwise the same as SGPC and maintains its guarantee of stability so long as the now constrained optimisation remains feasible.

2.1 The SGPC strategy

Let the system model be given in terms of the delay operator, z^{-1}, as:

$$G(z)=\frac{z^{-1}b(z)}{a(z)}=\frac{z^{-1}\left(b_0+b_1z^{-1}+b_2z^{-2}+\cdots+b_{n_b}z^{-n_b}\right)}{1+a_1z^{-1}+a_2z^{-2}+\cdots+a_{n_a}z^{-n_a}} \qquad \textbf{(2.1)}$$

where $a(z)$ and $b(z)$ are coprime and more delays can be incorporated by setting $b_i=0$ for $i=0,1,2,\ldots,k$. Form the stabilising feedback loop of Figure 2.1 such that the system output y, input increment Δu, input u, and commanding input c are related as:

$$y_t=z^{-1}b(z)c_t; \quad \Delta u_t=\alpha(z)c_t; \quad \Delta u_t=\Delta(z)u_t; \quad \alpha(z)=a(z)\Delta(z); \quad \Delta(z)=1-z^{-1} \qquad \textbf{(2.2)}$$

From Figure 2.1, the closed-loop transfer functions from c_t to y_t and from c_t to u_t are easily derived and are given as:

$$y_t = \frac{z^{-1}b(z)}{\alpha(z)M^{\#}(z) + z^{-1}b(z)N^{\#}(z)}c_t; \qquad u_t = \frac{a(z)}{\alpha(z)M^{\#}(z) + z^{-1}b(z)N^{\#}(z)}c_t \tag{2.3}$$

Thus, the controller numerator and denominator polynomials, $M^{\#}(z)$ and $N^{\#}(z)$, for which eqns. (2.2)a,b hold true, are defined by the Bezout identity [19]:

$$\alpha(z)M^{\#}(z) + z^{-1}b(z)N^{\#}(z) = 1 \tag{2.4}$$

Next define the Toeplitz convolution matrix, C_p, and the Hankel matrix, H_p, associated with the polynomial $p(z)=p_0+p_1z^{-1}+\cdots+p_{n_p}z^{-n_p}$ to be:

$$C_p = \begin{bmatrix} p_0 & 0 & \cdots & \vdots & \cdots & \cdots & 0 \\ p_1 & p_0 & \cdots & \vdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & 0 \\ p_{n_p} & p_{n_p-1} & \vdots & \vdots & p_0 & \vdots & 0 \\ 0 & p_{n_p} & p_{n_p-1} & \vdots & \vdots & p_0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \vdots & p_{n_p} & \vdots & \vdots & p_1 & p_0 \end{bmatrix} = [\Gamma_p \;\vdots\; \Theta_p]; \qquad H_p = \begin{bmatrix} p_1 & p_2 & \cdots & \cdots & p_{n_p} \\ p_2 & p_3 & \cdots & p_{n_p} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{n_p} & 0 & \cdots & \cdots & 0 \\ 0 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{2.5}$$

where the dimensions of C_p, H_p and of the partition matrices Γ_p, Θ_p of C_p will vary for the different choices of $p(z)$ and horizons, as will be obvious from the context; in the sequel the vector $\boldsymbol{\theta}_p$ will represent the sum of all the column vectors in Θ_p. Further, define the elements of the vectors of future and past outputs/input increments and future inputs/commanding inputs to be:

$$\underset{\rightarrow}{\boldsymbol{y}} = \begin{bmatrix} y_{t+1} \\ y_{t+2} \\ \vdots \\ y_{t+n} \end{bmatrix}; \; \underset{\leftarrow}{\boldsymbol{y}} = \begin{bmatrix} y_t \\ y_{t-1} \\ \vdots \\ y_{t-n_a} \end{bmatrix}; \; \underset{\rightarrow}{\Delta\boldsymbol{u}} = \begin{bmatrix} \Delta u_t \\ \Delta u_{t+1} \\ \vdots \\ \Delta u_{t+n-1} \end{bmatrix}; \; \underset{\leftarrow}{\Delta\boldsymbol{u}} = \begin{bmatrix} \Delta u_{t-1} \\ \Delta u_{t-2} \\ \vdots \\ \Delta u_{t-n_b} \end{bmatrix}; \; \underset{\rightarrow}{\boldsymbol{u}} = \begin{bmatrix} u_t \\ u_{t+1} \\ \vdots \\ u_{t+n-1} \end{bmatrix}; \; \underset{\rightarrow}{\boldsymbol{c}} = \begin{bmatrix} c_t \\ c_{t+1} \\ \vdots \\ c_{t+n-1} \end{bmatrix} \tag{2.6}$$

where n will be determined as needed.

Then, to derive the closed-loop output/input prediction equations for the stabilizing loop of Figure 2.1, we first write the relation of the current and past outputs to the current and past input increments in difference equation form:

$$\alpha(z)y_{t+1}=b(z)\Delta u_t \quad \Leftrightarrow \quad \alpha_0 y_{t+1}+\alpha_1 y_t+\cdots+\alpha_{n_a+1}y_{t-n_a}=b_0\Delta u_t+b_1\Delta u_{t-1}+\cdots+b_{n_b}\Delta u_{t-n_b} \tag{2.7}$$

Incrementing the subscripts on the outputs and input increments in eqn. (2.7)b by 1,2,…,n-1 yields n equations:

$$\begin{array}{llll}
\alpha_0 y_{t+1} & +\alpha_1 y_t+\cdots & +\alpha_{n_a+1}y_{t-n_a}=b_0\Delta u_t & +b_1\Delta u_{t-1}+\cdots+b_{n_b}\Delta u_{t-n_b} \\
\alpha_1 y_{t+1}+\alpha_0 y_{t+2} & +\alpha_2 y_t+\cdots+\alpha_{n_a+1}y_{t-n_a+1} & =b_1\Delta u_t+b_0\Delta u_{t+1} & +b_2\Delta u_{t-1}+\cdots+b_{n_b}\Delta u_{t-n_b+1} \\
\vdots & \vdots & \vdots & \vdots \\
\alpha_{n_a}y_{t+1}+\cdots+\alpha_0 y_{t+n_a+1} & +\alpha_{n_a+1}y_t & = b_{n_b}\Delta u_{t+n_a-n_b}+\cdots+b_0\Delta u_{t+n_a} & \\
\alpha_{n_a+1}y_{t+1}+\cdots+\alpha_0 y_{t+n_a+2} & & = b_{n_b}\Delta u_{t+n_a-n_b+1}+\cdots+b_0\Delta u_{t+n_a+1} & \\
\vdots & & \vdots & \\
\alpha_{n_a+1}y_{t+n-n_a-1}+\cdots+\alpha_0 y_{t+n} & & = b_{n_b}\Delta u_{t+n-n_b-1}+\cdots+b_0\Delta u_{t+n-1} &
\end{array} \tag{2.8}$$

which are written in matrix prediction equation form as:

$$C_\alpha \underset{\rightarrow}{y}+H_\alpha \underset{\leftarrow}{y}=C_b \underset{\rightarrow}{\Delta u}+H_b \underset{\leftarrow}{\Delta u} \tag{2.9}$$

Applying the same process to the relation of the current and past input increments to the command signal and the current and past outputs gives:

$$M^{\#}(z)\Delta u_t=c_t-z^{-1}N^{\#}(z)y_{t+1} \quad \Leftrightarrow \quad C_{M'}\underset{\rightarrow}{\Delta u}+H_{M'}\underset{\leftarrow}{\Delta u}=\underset{\rightarrow}{c}-C_{z^{-1}N'}\underset{\rightarrow}{y}-H_{z^{-1}N'}\underset{\leftarrow}{y} \tag{2.10}$$

Premultiplying eqn. (2.9) by $C_{M'}$ and eqn. (2.10)b by C_α, and utilizing the commutative property of convolution matrix multiplication, we get:

$$\begin{aligned}
C_\alpha C_{M'}\underset{\rightarrow}{y}+C_{M'}H_\alpha\underset{\leftarrow}{y}&=C_b C_{M'}\underset{\rightarrow}{\Delta u}+C_{M'}H_b\underset{\leftarrow}{\Delta u} \\
C_\alpha C_{M'}\underset{\rightarrow}{\Delta u}+C_\alpha H_{M'}\underset{\leftarrow}{\Delta u}&=C_\alpha\underset{\rightarrow}{c}-C_{z^{-1}N'}C_\alpha\underset{\rightarrow}{y}-C_\alpha H_{z^{-1}N'}\underset{\leftarrow}{y}
\end{aligned} \tag{2.11}$$

then, substituting eqn. (2.10)b (solved for $C_{M'}\underset{\rightarrow}{\Delta u}$) into (2.11)a and eqn. (2.9) (solved for $C_\alpha\underset{\rightarrow}{y}$) into (2.11)b, after some rearrangement, gives:

$$\begin{aligned}
\underset{\rightarrow}{y}&=(C_\alpha C_{M'}+C_b C_{z^{-1}N'})\ \underset{\rightarrow}{y}\ =C_b\underset{\rightarrow}{c}-L_1\underset{\leftarrow}{y}-L_2\underset{\leftarrow}{\Delta u}; & L_1&=C_b H_{z^{-1}N'}+C_{M'}H_\alpha \\
\underset{\rightarrow}{\Delta u}&=(C_\alpha C_{M'}+C_b C_{z^{-1}N'})\underset{\rightarrow}{\Delta u}=C_\alpha\underset{\rightarrow}{c}-L_3\underset{\leftarrow}{y}-L_4\underset{\leftarrow}{\Delta u}; & L_2&=C_b H_{M'}-C_{M'}H_b \\
& & L_3&=C_\alpha H_{z^{-1}N'}-C_{z^{-1}N'}H_\alpha \\
& & L_4&=C_\alpha H_{M'}+C_{z^{-1}N'}H_b
\end{aligned} \tag{2.12}$$

where the first equalities above are a direct consequence of bezout identity (2.4), in that

$(C_\alpha C_{M'} + C_b C_{z^{-1}N'}) = I$.

The SGPC strategy is to choose $c(z)$ such that the predicted y reaches the set-point r and Δu reaches zero in a finite number of steps. In particular, it is intended that $y_{t+i}=r$, $i>n_y$ and $\Delta u_{t+i-1}=0$, $i>n_u$; n_y and n_u are referred to as output and input horizons. This strategy is implemented by selecting $c(z)=c_t+c_{t+1}z^{-1}+\cdots+c_{t+n_c-1}z^{-n_c+1}+c_\infty z^{-n_c}/(1-z^{-1})$, where $c_\infty = r/b(1)$; n_c therefore denotes the number of control degrees of freedom and is referred to as the command horizon. Thus, the predicted y, Δu, u, c reach their steady-state values in n_y, n_u, n_u-1, n_c steps, and hence we need only concern ourselves with predicted values up to the appropriate horizons, therefore n in the prediction vectors of eqns. (2.6)a,c,e,f is assumed to be n_y, n_u, n_u, n_c. Eqns. (2.12) can then be rewritten as:

$$\underset{\rightarrow}{y}=\Gamma_b \underset{\rightarrow}{c}+\theta_b c_\infty+\underset{\rightarrow}{y_f}; \qquad \underset{\rightarrow}{\Delta u}=\Gamma_\alpha \underset{\rightarrow}{c}+\theta_\alpha c_\infty+\underset{\rightarrow}{\Delta u_f}; \qquad \underset{\rightarrow}{u}=\Gamma_a \underset{\rightarrow}{c}+\theta_a c_\infty+\underset{\rightarrow}{u_f} \tag{2.13}$$

where, as mentioned above, the scalar c_∞ is the desired steady-state value for the commanding input c of the system of Figure 2.1 and is chosen to remove steady-state offsets in the output predictions. Γ_b is $n_y \times n_c$; Γ_α, Γ_a are $n_u \times n_c$; and the vectors $\underset{\rightarrow}{y_f}$, $\underset{\rightarrow}{\Delta u_f}$, and $\underset{\rightarrow}{u_f}$ depend on past data (and account for non-zero initial conditions), are known, and are given as:

$$\underset{\rightarrow}{y_f}=-L_1\underset{\leftarrow}{y}-L_2\underset{\leftarrow}{\Delta u}; \qquad \underset{\rightarrow}{\Delta u_f}=-L_3\underset{\leftarrow}{y}-L_4\underset{\leftarrow}{\Delta u}; \qquad \underset{\rightarrow}{u_f}=-C_\Delta^{-1}(L_3\underset{\leftarrow}{y}+L_4\underset{\leftarrow}{\Delta u})+u_{t-1}\mathbf{1} \tag{2.14}$$

Eqn. (2.13)c is derived with the help of eqn. (2.2)c, which can be written in prediction equation form as:

$$\underset{\rightarrow}{u}=C_\Delta^{-1}(\underset{\rightarrow}{\Delta u}-H_\Delta u_{t-1})=C_\Delta^{-1}\underset{\rightarrow}{\Delta u}+u_{t-1}\mathbf{1} \tag{2.15}$$

C_Δ^{-1} is a lower triangular matrix of ones and $\mathbf{1}$ is a vector of ones.

The SGPC optimization is then defined through the minimisation of a standard

GPC cost:

$$J = \sum_{i=1}^{n_y} (r_{t+i} - y_{t+i})^2 + \lambda \sum_{i=0}^{n_u-1} \Delta u_{t+i}^2 = \left\| \underset{\rightarrow}{r} - \underset{\rightarrow}{y} \right\|_2^2 + \lambda \left\| \underset{\rightarrow}{\Delta u} \right\|_2^2 = [\underset{\rightarrow}{c} - \underset{\rightarrow}{c}_o]^T S^2 [\underset{\rightarrow}{c} - \underset{\rightarrow}{c}_o] + \gamma \qquad \textbf{(2.16)}$$

where, for the last equality, we have substituted for $\underset{\rightarrow}{y}$, $\underset{\rightarrow}{\Delta u}$ from eqns. (2.13)a,b; $\underset{\rightarrow}{r} = [r_{t+1}, r_{t+2}, \ldots, r_{t+n_r}, \ldots, r_{t+n_r}]^T \in \Re^{n_y}$ is the vector of future set-points, with n_r the reference horizon and r_{t+n_r} referred to in the text as r; and the matrix S^2, the optimum vector of future command inputs $\underset{\rightarrow}{c}_o$, and the optimum cost γ are given as:

$$S^2 = \Gamma_b^T \Gamma_b + \lambda \Gamma_\alpha^T \Gamma_\alpha; \qquad \underset{\rightarrow}{c}_o = S^{-2} [\Gamma_b^T (\underset{\rightarrow}{r} - \underset{\rightarrow}{y}_f - \theta_b c_\infty) - \lambda \Gamma_\alpha^T (\underset{\rightarrow}{\Delta u}_f + \theta_\alpha c_\infty)];$$
$$\gamma = \left\| \underset{\rightarrow}{r} - \underset{\rightarrow}{y}_f - \theta_b c_\infty \right\|^2 + \lambda \left\| \underset{\rightarrow}{\Delta u}_f + \theta_\alpha c_\infty \right\|^2 - \left\| S \underset{\rightarrow}{c}_o \right\|^2 \qquad \textbf{(2.17)}$$

As there are no constraints, J is minimized by $\underset{\rightarrow}{c} = \underset{\rightarrow}{c}_o$ and the optimum current control increment, Δu_t, is computed as the first element of $\underset{\rightarrow}{\Delta u}$ of eqn. (2.13)b and implemented; this procedure is then repeated at the next sampling instant.

This strategy mirrors that of GPC with one important difference: in GPC the free variable Δu is related to y through the infinite impulse response of $G(z)$, whereas here the free variable c is related to y and Δu through the finite length sequences of eqns. (2.2)a,b. Therefore, if $n_y \geq n_b + n_c$ and $n_u \geq n_a + n_c + 1$, then the cost J is equivalent to an infinite horizon cost:

$$J = \sum_{i=1}^{n_y} (r_{t+i} - y_{t+i})^2 + \lambda \sum_{i=0}^{n_u-1} \Delta u_{t+i}^2 = \sum_{i=1}^{\infty} (r_{t+i} - y_{t+i})^2 + \lambda \sum_{i=0}^{\infty} \Delta u_{t+i}^2 \qquad \textbf{(2.18)}$$

and it is shown below that this cost decreases monotonically over time and thus constitutes a stable Lyapunov function, guaranteeing the stability of SGPC; hereafter, we assume that $n_y \geq n_b + n_c$ and $n_u \geq n_a + n_c + 1$.

Theorem 2.1 [19] SGPC is stable and gives asymptotic tracking.

Proof: Assume, without loss of generality, that the reference is fixed at r. Let the optimum sequence of future input increments Δu_{t+i} at sample t (ie. the elements of $\vec{\Delta u}$ of eqn. (2.13)b calculated with the optimum vector of future command inputs $\vec{c}_o$) give a cost J_t. Move forward to sample instant $t+1$, and let $J_{t+1|t}$ be the cost of using the same sequence of input increments (bar the first which is now in the past), then we have:

$$J_{t+1|t} = J_t - (r_{t+1} - y_{t+1})^2 - \lambda \Delta u_t^2 \tag{2.19}$$

Clearly, the optimum J_{t+1} is always such that $J_{t+1} \leq J_{t+1|t} \leq J_t$ (the first inequality is due to the extra degree of freedom at $t+1$). Equality can persist if, and only if, $r=y$ and $\Delta u=0$; thus, the cost J is a stable Lyapunov function, and SGPC is stable and gives asymptotic tracking. □

Remark 2.1 In the paragraph prior to Theorem 2.1 and throughout this thesis, we employ a slight abuse of terminology. We shall use the term, finite length sequence (FLS), to imply, not only sequences which are finite in length, but also those which settle at some steady-state value within finite time (eg. the predicted values of y, which are forced to settle at r after n_y steps). Later, we shall use the term, infinite length sequence (ILS), to imply sequences which reach a given value only asymptotically.

2.2 Robustness analysis and optimization for SGPC

When not subject to input constraints, the cost J has the explicit optimum solution $\vec{c}_o$ of eqn. (2.17)b. Thus, SGPC can be implemented with a fixed-term feedback controller; the derivation is as follows.

Substituting eqns. (2.14)a,b and $c_\infty=[\mathbf{0}^T\ 1/b(1)]\vec{\boldsymbol{r}}$, where $\mathbf{0}$ is a vector of zeros, into eqn. (2.17)b and then pre-multiplying by $\boldsymbol{e}_1$, the first standard basis vector, gives:

$$\begin{aligned} c_t &= \boldsymbol{p}_r\vec{\boldsymbol{r}}+\boldsymbol{p}_y\overleftarrow{\boldsymbol{y}}+\boldsymbol{p}_u\overleftarrow{\Delta\boldsymbol{u}} \\ \boldsymbol{p}_r &= \boldsymbol{e}_1^T S^{-2}\Gamma_b^T-[\mathbf{0}^T\ \boldsymbol{e}_1^T S^{-2}(\Gamma_b^T\theta_b+\lambda\Gamma_\alpha^T\theta_\alpha)/b(1)] \\ \boldsymbol{p}_y &= \boldsymbol{e}_1^T S^{-2}(\Gamma_b^T L_1+\lambda\Gamma_\alpha^T L_3) \\ \boldsymbol{p}_u &= \boldsymbol{e}_1^T S^{-2}(\Gamma_b^T L_2+\lambda\Gamma_\alpha^T L_4) \end{aligned} \qquad \textbf{(2.20)}$$

Let the coefficients of the anti-causal polynomial (ie. with positive powers of z), $p_r(z)$, and the causal polynomials, $p_y(z)$ and $p_u(z)$, be the elements of the vectors, $\boldsymbol{p}_r$, $\boldsymbol{p}_y$, and $\boldsymbol{p}_u$; then we may write:

$$c_t = p_r(z)r_{t+1}+p_y(z)y_t+p_u(z)\Delta u_{t-1} = p_r(z)r_{t+1}+p_y(z)y_t+z^{-1}p_u(z)\Delta u_t \qquad \textbf{(2.21)}$$

Substitution of this into eqn. (2.10)a gives:

$$M^{\#}(z)\Delta u_t = p_r(z)r_{t+1}+p_y(z)y_t+z^{-1}p_u(z)\Delta u_t-N^{\#}(z)y_t \qquad \textbf{(2.22)}$$

$$\Leftrightarrow$$

$$D_k(z)\Delta u_t = p_r(z)r_{t+1}-N_k(z)y_t; \qquad D_k(z)=M^{\#}(z)-z^{-1}p_u(z), \qquad N_k(z)=N^{\#}(z)-p_y(z)$$

Then SGPC can be implemented with a fixed term controller as shown in Figure 2.2. Thus, the closed-loop transfer functions from r to y and from r to u are given as:

$$\frac{y(z)}{r(z)}=\frac{z^{-1}b(z)p_r(z)}{p_c(z)}; \quad \frac{u(z)}{r(z)}=\frac{a(z)p_r(z)}{p_c(z)}; \quad p_c(z)=a(z)\Delta(z)D_k(z)+z^{-1}b(z)N_k(z) \qquad \textbf{(2.23)}$$

Now, the controller numerator and denominator polynomials, $\Delta(z)D_k(z)$ and $N_k(z)$, appear only in the closed-loop pole polynomial, $p_c(z)$, thus, *any* controller numerator and denominator polynomials, say $\Delta(z)D(z)$ and $N(z)$, which satisfy diophantine equation (2.23)c, will be optimal with respect to the cost J. The entire class of such controllers is given as [19]:

$$D(z)=D_k(z)-z^{-1}b(z)Q(z); \qquad N(z)=N_k(z)+\alpha(z)Q(z) \qquad \textbf{(2.24)}$$

where $Q(z)$ is an arbitrary stable transfer function.

The appropriate transfer function for analysing both additive model uncertainty

and the effects of feedback noise on the plant input is:

$$K(z)S(z) = \frac{N(z)}{\Delta(z)D(z)} \frac{a(z)\Delta(z)D(z)}{p_c(z)} = \frac{a(z)N(z)}{p_c(z)} = \frac{a(z)(N_k(z) + \alpha(z)Q(z))}{p_c(z)} \qquad \textbf{(2.25)}$$

where $S(z)$ is the sensitivity transfer function. Thus, the problem is decoupled such that the degrees of freedom available in $Q(z)$ can be deployed to improve robustness and/or noise handling properties *without* affecting optimality with respect to J (ie. performance).

Consider, for instance, robustness. If the size of the additive model uncertainty is bounded by the modulus of a stable function, $W(z)$, then the necessary and sufficient condition for robust stability is:

$$\| W(z)K(z)S(z) \|_\infty \leq 1; \qquad z=e^{j\theta}, \quad 0 \leq \theta < \pi \qquad \textbf{(2.26)}$$

Combining this with eqn. (2.25) implies that the optimal $Q(z)$, with respect to stability robustness, is given as the solution to the following standard H_∞ optimization problem:

$$\min_{Q(z)} \| T_1(z) - T_2(z)Q(z) \|_\infty; \quad T_1(z) = \frac{W(z)a(z)N_k(z)}{p_c(z)}, \quad T_2(z) = \frac{W(z)a(z)\alpha(z)}{p_c(z)} \qquad \textbf{(2.27)}$$

The solution to this problem is well documented, and we note that if $Q(z)$ is restricted to finite length polynomials of degree n_Q, then the optimal choice of such a $Q(z)$ can be determined using Lawson's weighted least squares algorithm [19].

2.3 Introducing constraints into SGPC

In most practical applications there are constraints on the values the system inputs are allowed to take; in this section we describe how these constraints are included in SGPC. For brevity, we consider only input rate and absolute constraints of the form:

$$\left.\begin{matrix} |\Delta u_{t+i}| \le R \\ |u_{t+i} - U_o| \le U \end{matrix}\right\} i=0,1,\ldots,n_u-1 \quad \Leftrightarrow \quad \left\| \begin{bmatrix} \frac{1}{R}\underset{\rightarrow}{\Delta u} \\ \frac{1}{U}\left(\underset{\rightarrow}{u} - U_o \underset{\rightarrow}{1}\right) \end{bmatrix} \right\|_\infty \le 1; \quad \underset{\rightarrow}{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in R^{n_u} \qquad \textbf{(2.28)}$$

On account of the FLS relationship of eqn. (2.2)b, the form of $c(z)$, and the condition that $n_u \ge n_a + n_c + 1$, the predicted inputs reach their steady-state values in n_u steps; hence, constraints (2.28)a need only be invoked over the input horizon n_u and are automatically satisfied for $i \ge n_u$. Using eqns. (2.13)b,c, we may rewrite constraints (2.28)b as:

$$\left\| A\underset{\rightarrow}{c} - \underset{\rightarrow}{v}(t) \right\|_\infty \le 1; \qquad A = \begin{bmatrix} \frac{1}{R}\Gamma_\alpha \\ \frac{1}{U}\Gamma_a \end{bmatrix} \qquad \underset{\rightarrow}{v}(t) = \begin{bmatrix} -\frac{1}{R}\underset{\rightarrow}{\Delta u_f} \\ \frac{1}{U}\left(U_o \underset{\rightarrow}{1} - \underset{\rightarrow}{u_f}\right) \end{bmatrix} - \begin{bmatrix} \frac{1}{R}\theta_\alpha c_\infty \\ \frac{1}{U}\theta_a c_\infty \end{bmatrix} \qquad \textbf{(2.29)}$$

where v is time dependent even if the constraints are time independent. Next, define the "feasible region", $F_{n_c}[v(t)]$, as the subspace of all n_c-dimensional vectors $\underset{\rightarrow}{c}$ which satisfy constraint (2.29)a. If $F_{n_c}[v(t)]$ is non-empty/empty for *all* n_c then the problem is said to be "long term" feasible/infeasible (LTF/LTIF); the terms "short term" feasible/infeasible (STF/STIF) will be used when $F_{n_c}[v(t)]$ is non-empty/empty for the *particular* chosen value of n_c. An LTIF problem cannot be controlled; thus, we will always assume LTF.

In the case of STIF, $\underset{\rightarrow}{c}$ should be chosen to minimize the worst case constraint violation, ie minimize $\left\| A\underset{\rightarrow}{c} - \underset{\rightarrow}{v}(t) \right\|_\infty$; this can be accomplished with Lawson's weighted least squares algorithm [22]. However, when $\left\| A\underset{\rightarrow}{c} - \underset{\rightarrow}{v}(t) \right\|_\infty$ can be made less than 1, the strategy for choosing c should be dominated by the minimization of the cost of eqn. (2.16). These two aims are combined in the mixed weight least squares (MWLS) iteration defined by:

MWLS:

Step 0: Initialize, $i=0$, $w^{(0)}=1$, $W^{(0)}=I_{r_A}/r_A$, where r_A is the row dimension of A.

Step 1: Increment i by 1 and minimize with respect to $\underset{\rightarrow}{c}$ the cost:

$$J_{MWLS}^{(i)}=\left\|\begin{bmatrix}[w^{(i)}]^{1/2}\epsilon^{(i)}\\ [W^{(i)}]^{1/2}e^{(i)}\end{bmatrix}\right\|_2^2; \qquad \epsilon^{(i)}=S[\underset{\rightarrow}{c}-\underset{\rightarrow}{c}_o]; \qquad e^{(i)}=A\underset{\rightarrow}{c}-v(t) \tag{2.30}$$

Step 2: If the change in cost is less than a preset threshold quit, otherwise update the weights according to the equations below and go to Step 1:

$$w^{(i+1)}=\frac{w^{(i)}}{\|W^{(i)}e^{(i)}\|_1}; \qquad W_{jj}^{(i+1)}=\frac{W_{jj}^{(i)}|e^{(i)}{}_j|}{\|W^{(i)}e^{(i)}\|_1} \tag{2.31}$$

MWLS has some desirable properties [33]: under short term feasibility (STF) it can only converge to the constrained optimum, $\underset{\rightarrow}{c}^*$; in the case of STIF it converges to the solution that minimizes the maximum constraint violation. The latter constitutes an important feature: it prescribes one way of handling STIF. This strategy may not always be the best; however, by minimizing the predicted maximum (and possibly future) constraint violation, the algorithm allows for the possibility of avoiding violations altogether.

The following algorithm implements CSGPC:

Algorithm 2.1 (CSGPC)

Step 1: Calculate the vector of future command inputs which minimizes the two-norm of the predicted errors and weighted control increments without violating the constraints, namely, $\min_c J_t=\{\|\epsilon\|_2^2$ s.t. $\|A\underset{\rightarrow}{c}-v(t)\|_\infty\le 1$, $c_\infty=r/b(1)\}$.

Step 2: Calculate and implement the first control increment using eqn. (2.13)b.

Step 3: Increment t by one and return to Step 1.

CSGPC, like SGPC, has an attendant stability theory which is established by proving that the relevant cost is a monotonically decreasing function of time [33]. For CSGPC however, this requires the assumption that the problem remains STF as stated in the theorem below:

Theorem 2.2 [33] Let a linear system with transfer function $G(z)=z^{-1}b(z)/a(z)$ be subject to input constraints which, at time t and for horizon n_c, are given as $\|A\vec{c}-v(t)\|_\infty \leq 1$; and let $F_{n_c}[v(t)]$ denote the implied feasible region for c. Then if $F_{n_c}[v(t)]$ is non-empty for all t, CSGPC will cause y to follow asymptotically any set-point change.

Proof: Assuming feasibility of the now constrained optimization, the arguments used in the proof of Theorem 2.1 still hold. □

The feasibility assumption of Theorem 2.2 involves short term feasibility (STF), and thus the assumption is a strong condition. The purpose of this work is to find ways to ensure its satisfaction and to make it as weak as possible; in next chapter we first derive tests for feasibility and then conditions which must be satisfied to ensure feasibility and thus stability for all future times. Chapter 4 then deals with large set-point changes as a cause of STIF, Chapter 5 with making the STF assumption as weak as possible, and Chapter 6 with disturbances as a cause of STIF.

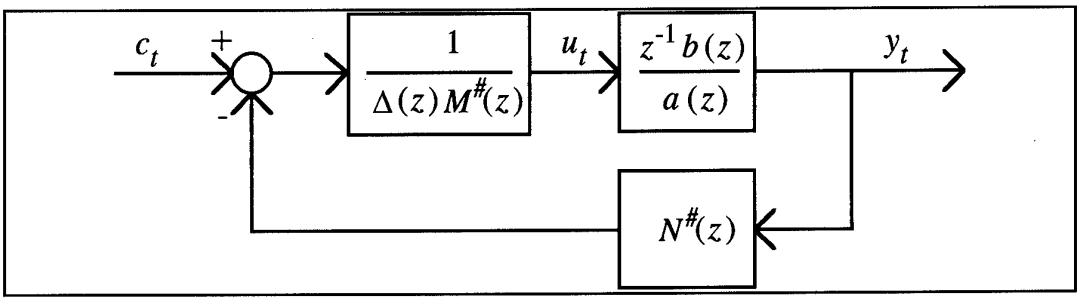


Figure 2.1 The stabilizing feedback loop

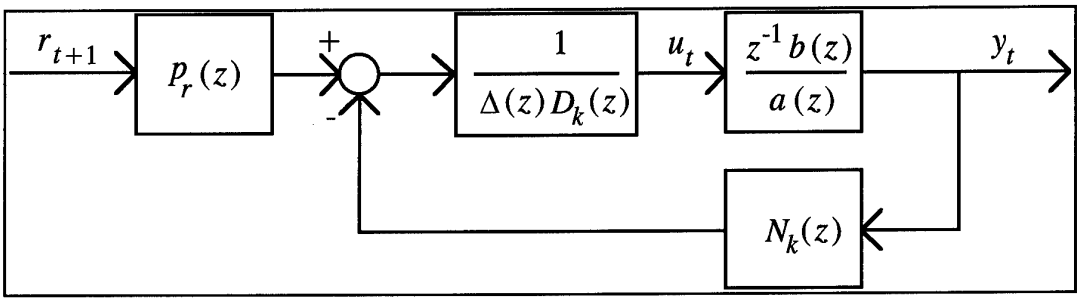


Figure 2.2 Optimized SGPC feedback loop

Chapter 3

Feasibility and Stability for Constrained Control

The feasibility assumption of Theorem 2.2 is strong because it implies "short term" feasibility (STF), namely the feasibility of making the output reach the desired set-point in a finite number of steps without violating the constraints. The fact that a system output can eventually reach a target value without violating constraints (long term feasibility, or LTF) does not imply it can do so in n_y steps and with only n_c command input changes (STF). CSGPC, like other constrained predictive control algorithms, can, in fact, destabilize what was originally an LTF problem; by requiring the predicted output to reach its target within n_y steps (a terminal constraint), it may be necessary to drive the controls and/or their increments at their limits. If, however, a system is unstable and/or non-minimum phase, these earlier control moves may require future stabilizing control moves which will not be available within the existing input constraints; this will lead to instability.

As only the first input, u_t, is actually implemented, when dealing with open-loop unstable systems which are subject to input constraints, one must ask what restrictions should be placed on this input to ensure a continued guarantee of stability and feasibility.

In Section 3.1, explicit stability conditions are developed for systems with one or two real unstable poles; the conditions are first imposed on past data (*a posteriori* conditions), and then on u_t itself (*a priori* conditions). In Section 3.2, we show that linear programming can be used to provide *a priori* stability results for systems with any number of (real and/or complex) unstable poles. We concentrate on finding conditions which are both necessary and sufficient to ensure stability and thus regain some of the degrees of freedom given up by algorithms like those in ([33], [30], [54], [18]) which enforce overly stringent terminal constraints on the predicted outputs/inputs; relaxing these constraints implies that feasibility can be retained with shorter command horizons which afford significant reductions in the computational load. A numerical example is given which shows that these stability conditions can be used in a supervisory role with algorithms which have no guarantee of stability.

3.1 Systems with one or two real poles

In this section we develop necessary and sufficient conditions under which instability can be avoided. Section 3.1.1 sets up the conditions which are needed to determine feasibility and stability. Then, section 3.1.2 gives the necessary and sufficient conditions for *a posteriori* stability; these conditions are *a posteriori* in that they are imposed on past data. If *a posteriori* conditions are violated, instability will be the result, but this only provides a useful test for when things are about to go wrong. In Section 3.1.3, we consider the problem of propagating these stability conditions forward in time and thus propose a procedure for the derivation of *a priori* conditions which ensure the continuing existence of a stabilizing solution. Obviously, any algorithm which ignores

these conditions and violates them will be unstable. In chapter 6, we will demonstrate the use of these *a priori* conditions in the presence of disturbances.

3.1.1 Setting up the conditions for feasibility with stability

We first develop the mathematical tools needed, then state the conditions for feasibility and stability.

3.1.1.1 Mathematical preliminaries

Consider the lower triangular $n_u \times n_u$ toeplitz convolution matrix formed of the coefficients of $a(z)$, C_a, as defined in eqn. (2.5). Then we have the following result:

Lemma 3.1 Let σ_i be the singular values of C_a and let $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{n_u}$. Then if $a(z)=1-pz^{-1}$ with $|p| > 1$ and $q=1/p$:

$$(|p|-1) \leq \sigma_{n_u-1} \leq \sigma_{n_u-2} \leq \cdots \leq \sigma_1 \leq (|p|+1); \quad \text{and} \quad \sigma_{n_u} < |q|^{n_u-1} \tag{3.1}$$

Proof: Condition (3.1)a follows from an application of Gershgoreen's theorem to the matrix $C_a^T C_a$ which has a tridiagonal form with n_u-1 diagonal elements equal to p^2+1 and one diagonal element equal to 1, whereas all the non-zero off diagonal elements are equal to p. To complete the proof, post-multiply C_a by the vector $[q^{n_u-1}, q^{n_u-2}, \ldots, 1]^T$ to get the first standard vector multiplied by q^{n_u-1}. Then invoking a norm inequality we get $\sigma_{n_u}(1-q^{2n_u})/(1-q^2) \leq q^{n_u-1}$, which leads to condition (3.1)b. □

We can then show that the output principle direction associated with σ_{n_u} (and hence the input direction associated with the singular value of C_a^{-1} which is "blowing up") has an

easily determined form:

Theorem 3.1 With the definitions of Lemma 3.1 let $\boldsymbol{k}_i$ be the principal input directions of $\boldsymbol{q}^T=[1, q, q^2, ..., q^{n_u-1}]$, and $\boldsymbol{k}_{n_u}$ be the direction associated with the non-zero singular value of $\boldsymbol{q}^T$. Then writing any $\boldsymbol{w}$ in R^{n_u} as $\boldsymbol{w}=K\boldsymbol{\alpha}$, for $K=[\boldsymbol{k}_1, \boldsymbol{k}_2, ..., \boldsymbol{k}_{n_u}]$, $\boldsymbol{\alpha}=[\alpha_1, \alpha_2, ..., \alpha_{n_u}]^T$, we have

$$\| C_a^{-1}\boldsymbol{w} \|_2 \le \frac{1}{|p|-1} \| \boldsymbol{\alpha} \|_2 \quad \text{for} \quad \alpha_{n_u}=0; \quad \| C_a^{-1}\boldsymbol{w} \|_2 \ge |\alpha_{n_u}|\,|p|^{n_u-1} \quad \text{for} \quad \alpha_{n_u} \neq 0 \tag{3.2}$$

Proof: Let $[\boldsymbol{x}_i, \sigma_i, \boldsymbol{y}_i]$ denote the output principal directions, singular values, and input principal directions of C_a. Then by Lemma 3.1, it is easy to show that $\boldsymbol{q}^T/\| \boldsymbol{q}^T \|_2 = \boldsymbol{k}_{n_u} = \boldsymbol{x}_{n_u}$, so that $\boldsymbol{k}_i$ can always be chosen such that $\boldsymbol{k}_i = \boldsymbol{x}_i$. Then given that $[\boldsymbol{y}_i, 1/\sigma_i, \boldsymbol{x}_i]$ denotes the singular value/vector triple for C_a^{-1}, it follows that

$$\| C_a^{-1}\boldsymbol{w} \|^2 = \| \sum_{i=1}^{n_u} (\alpha_i / \sigma_i)\boldsymbol{y}_i \|^2 = \sum_{i=1}^{n_u} (\alpha_i / \sigma_i)^2 \tag{3.3}$$

The result follows by invoking Lemma 3.1 for $\alpha_{n_u}=0$ and $\alpha_{n_u} \neq 0$. □

The condition for keeping $C_a^{-1}\boldsymbol{w}$ finite, then, is that $\boldsymbol{w}$ must be orthogonal to $\boldsymbol{q}^T$. Next, we expand the development to consider systems with more than one unstable pole.

Lemma 3.2 Let $a_i(z)=1-p_i z^{-1}$, with a corresponding convolution matrix C_{ai}, for $i=1,2,\ldots,n$ and let $a(z)=a_1(z)\times\ldots\times a_n(z)$. Then C_a, the convolution matrix for $a(z)$, is the product of the C_{ai} taken in any order.

Proof: This is straightforward and, for brevity, will be omitted. □

Theorem 3.2 Let the roots of $a(z)$ be p_i, let $|p_i|>1$ for $i=1,2,\ldots,m$ and let the $m\times n_u$ matrix Q have as its i^{th} row the vector $\boldsymbol{q}_i^T=[1,q_i,q_i^2,\ldots,q_i^{n_u-1}]$, where $q_i=1/p_i$. Furthermore, denote by $\boldsymbol{k}_i$ the input principal directions of Q arranged so that the $\boldsymbol{k}_i$ for $i=n_u-m+1,n_u-m+2,\ldots,n_u$ correspond to the non-zero singular values of Q. Then, expressing any vector $\boldsymbol{w}$ in R^{n_u} as $\boldsymbol{w}=K\boldsymbol{\alpha}+K'\boldsymbol{\alpha}'$ where $K=[\boldsymbol{k}_1,\boldsymbol{k}_2,\ldots,\boldsymbol{k}_{n_u-m}]$ is the kernel of Q and $K'=[\boldsymbol{k}_{n_u-m+1},\boldsymbol{k}_{n_u-m+2},\ldots,\boldsymbol{k}_{n_u}]$ is the row space, we have, for $j=1,\ldots,m$, that

$$\|C_a^{-1}\boldsymbol{w}\|_2\le\prod_{i=1}^{n_u-m}\left(\frac{1}{|p_i|-1}\right)\|\boldsymbol{\alpha}\|_2,\quad \boldsymbol{\alpha}'=\boldsymbol{0};\qquad \|C_a^{-1}\boldsymbol{w}\|_2\ge O(p_j^{n_u-1}),\quad \boldsymbol{\alpha}'\neq\boldsymbol{0} \tag{3.4}$$

Proof: Let $m=2$ so that $C_a=C_{a1}C_{a2}$, let K_1, K_2 be matrix representations of the kernel of the rows of Q, $\boldsymbol{q}_1^T$, $\boldsymbol{q}_2^T$, defined in a manner analogous to K, and let $C_{a1}^{-1}\boldsymbol{w}=\boldsymbol{v}$. Then $\boldsymbol{\alpha}'=\boldsymbol{0}$, together with the fact that the last row of C_{a1}^{-1} is $p_1^{n_u-1}[1,q_1,q_1^2,\ldots,q_1^{n_u-1}]$, imply that the last element of $\boldsymbol{v}$ is zero, so that $\boldsymbol{w}=C_{a1}\boldsymbol{v}$ gives a matrix representation of the convolution implied by $w(z)=(1-p_1z^{-1})v(z)$. But, with $\boldsymbol{\alpha}'=\boldsymbol{0}$, $Q\boldsymbol{w}=\boldsymbol{0}$, which implies that $w(p_1)=w(p_2)=0$, so $w(z)=(1-p_1z^{-1})(1-p_2z^{-1})\phi(z)$, where $\phi(z)$ is of order n_u-m-2, and thus $v(z)=(1-p_2z^{-1})\phi(z)$ or $v(p_2)=0$; hence $\boldsymbol{q}_2^T\boldsymbol{v}=0$, or $\boldsymbol{v}=K_2\boldsymbol{\beta}$ for some non-zero vector $\boldsymbol{\beta}$. Then, by Theorem 3.1, we have

$$\|C_{a2}^{-1}\boldsymbol{v}\|\ \le\ \|\boldsymbol{\beta}\|/(|p_2|-1);\qquad \|\boldsymbol{v}\|=\|C_{a1}^{-1}\boldsymbol{w}\|\ \le\ \|\boldsymbol{\gamma}\|/(|p_1|-1) \tag{3.5}$$

where $\boldsymbol{\gamma}$ is the projection of $\boldsymbol{w}$ onto the kernel of $\boldsymbol{q}_1^T$. However $\boldsymbol{w}=K\boldsymbol{\alpha}=K_1\boldsymbol{\gamma}$ and $\boldsymbol{v}=K_2\boldsymbol{\beta}$, so that $\|\boldsymbol{\gamma}\|=\|\boldsymbol{\alpha}\|$ and $\|\boldsymbol{\beta}\|=\|\boldsymbol{v}\|$ which combine with conditions (3.5)a,b to give condition (3.4)a for $m=2$; these arguments apply for a general m. Conversely, if at least one of the elements of $\boldsymbol{\alpha}'$, say the j^{th}, is non-zero then, by Theorem 3.1, $\|C_{aj}^{-1}\boldsymbol{w}\|_2\ge O(p_j^{n_u-1})$. □

3.1.1.2 Necessary and sufficient conditions for feasibility and stability

In the absence of constraints, SGPC has guaranteed stability because u is chosen so that the predicted y reaches its set-point within a finite number of steps; this implicitly involves the "pseudo" cancellation of all the poles of $a(z)$ [34]. However, when u is constrained and we encounter short term infeasibility, this property is lost, and u does not "cancel" the unstable poles of $a(z)$; then not only will the predicted y not reach its steady-state value, but the control system will go unstable. Below we state the necessary and sufficient conditions which avoid this situation.

Remark 3.1 By pseudo cancellation, we imply that the future values of u are chosen such that, *taken in combination with the effects of past values of inputs and outputs*, the effects of the roots of the denominator polynomial are "cancelled". Thus, we refer only to "cancellation" in a closed-loop feedback sense, like that of Figure 2.1, and do not refer to a direct controller-plant cancellation which, for unstable cancellations, has inherent internal stability problems. Hereafter, we shall continue to refer to this differentiation by using the word pseudo or placing the word, cancel, in quotes.

As was done in Section 2.1 to generate the future values of y and Δu from eqn. (2.7)a, the future values of y and u can be generated by simulating $a(z)y_{t+1}=b(z)u_t$ forward in time to get:

$$C_a \underset{\rightarrow}{y} = C_b \underset{\rightarrow}{u} - H_a \underset{\leftarrow}{y} + H_b \underset{\leftarrow}{u}; \quad \text{or} \quad \underset{\rightarrow}{y} = C_a^{-1} w; \quad \text{where} \quad w = C_b \underset{\rightarrow}{u} - H_a \underset{\leftarrow}{y} + H_b \underset{\leftarrow}{u} \qquad \textbf{(3.6)}$$

Note now, that due to the FLSs used in CSGPC, short term feasibility implies LTF, namely if $\| Ac\text{-}v(t) \|_\infty \leq 1$ for some finite n_c, then this inequality will also hold true as n_c

tends to infinity. Thus, determining necessary and sufficient conditions for feasibility (with stability) is equivalent to looking for the necessary and sufficient conditions under which, for $n_u \to \infty$, there exists a w such that $\underset{\rightarrow}{y}$ of eqn. (3.6)b is bounded and such that the vector of future u's defined by eqn. (3.6)c satisfies the input constraints.

Theorem 3.3 Let u be subject to constraints (2.28), let the poles p_i of $a(z)$ be such that $|p_j| > 1$ for $j=1,2,\ldots,m$ and let the matrix Q be defined as per Theorem 3.2. Then at time t, the problem is feasible if, and only if, for $n_u \to \infty$, there exists a $\underset{\rightarrow}{u}$ such that

$$\begin{aligned} &\underset{\rightarrow}{Qu} = \underset{u}{b} \qquad \text{and} \qquad \| C^u \underset{\rightarrow}{u} - u^o \|_\infty \le 1; \qquad \text{where} \\ &\underset{u}{b} = D_{b(p)}^{-1} Q(H_a \underset{\leftarrow}{y} - H_b \underset{\leftarrow}{u}); \quad C^u = [\frac{1}{U} I_{n_u}, \ \frac{1}{R} C_\Delta^T]^T; \quad u^o = [\frac{U_o}{U} \mathbf{1}^T, \ \frac{u_{t-1}}{R} e_1^T]^T \end{aligned} \tag{3.7}$$

where $D_{b(p)}$ is a diagonal matrix with the i^{th} diagonal element equal to $b(p_i)$ $i=1,2,\ldots,m$, $b(p_i)$ is the polynomial $b(z)$ evaluated at p_i, e_1 is the first column of I_{n_u} and C_Δ is as defined in eqn. (2.5)a for $\Delta(z)=1-z^{-1}$.

Proof: Eqns. (3.7)a,c require w to be orthogonal to Q, where, from the definition of C_b and with Q, C_b of infinite dimension, we have $QC_b = D_{b(p)}Q$. Satisfaction of this equality constraint, by Theorem 3.2, ensures that $\underset{\rightarrow}{y}$ of eqn. (3.6)b will be bounded; (3.7)b in conjunction with eqns. (3.7)d,e is a representation of constraints (2.28). □

The results above appear to be of limited practical use because of the requirement that $n_u \to \infty$. However, given stability, the future values of u will settle at some constant value u_∞ after say n_u-1 time instants. In the light of this remark, Theorem 3.3 may be restated as follows.

Theorem 3.4 Let u be subject to the constraints of Theorem 3.3. Then the necessary and sufficient condition for feasibility (and stability) is that for some $n_u > n_a$ there exists a $\underset{\rightarrow}{u}$ satisfying conditions (3.7), providing that the i^{th} element of the last column of Q is replaced by $q_i^{n_u-1}/(1-q_i)$ for $i=1,2,\ldots,m$.

Proof: Let all the future values of u from the $(n_u\text{-}1)^{\text{th}}$ time instant onwards be u_∞, then

$$\sum_{j=1}^{\infty} q_i^{j-1} u_{t+j-1} = \sum_{j=1}^{n_u-1} q_i^{j-1} u_{t+j-1} + \frac{q_i^{n_u-1}}{1-q_i} u_\infty \qquad \textbf{(3.8)}$$

and thus conditions (3.7) may be restated for n_u finite, so long as the last column of Q is replaced by the vector having as its i^{th} element the ratio $q_i^{n_u-1}/(1-q_i)$. Conditions (3.7) will then be a sufficient condition for feasibility, since it will guarantee the existence of a particular $\underset{\rightarrow}{u}$ which both satisfies the constraints and results in a stable $\underset{\rightarrow}{y}$. However, since n_u is allowed to be as large as needs be, the condition is also necessary. □

All conditions above involve linear equality and inequality constraints and thus in general provide no explicit results: (i) to test feasibility given past data (*a posteriori* feasibility); and (ii) given feasibility at t, to derive conditions on u_t that preserve feasibility and thus stability (*a priori* stability). *A posteriori* feasibility is dealt with in the next section; *a priori* stability will then be addressed in section 3.1.3.

3.1.2 Necessary and sufficient conditions for *a posteriori* feasibility

We first state a general test for feasibility and then develop explicit necessary and sufficient conditions for *a posteriori* feasibility with stability. Finally, we present two numerical examples which illustrate the results.

3.1.2.1 The feasibility conditions and the algorithm for testing feasibility

Theorem 3.4 implies that the problem of investigating feasibility boils down to that of finding whether an n_u exists for which conditions (3.7) have a solution $\underset{\rightarrow}{u}$. This can be tested easily, as stated below.

Theorem 3.5 Let K be a matrix representation of the kernel of Q, then there exists a vector $\underset{\rightarrow}{u}$ which satisfies conditions (3.7) if, and only if,

$$\inf_{\alpha} \| M'\alpha - \nu' \|_\infty \leq 1; \quad \text{where} \quad M' = C^u K; \quad \nu' = u^o - C^u Q^T (QQ^T)^{-1} b_u \quad \textbf{(3.9)}$$

Proof: Eqn. (3.7)a is satisfied for $\underset{\rightarrow}{u} = K\alpha + Q^T(QQ^T)^{-1}b_u$, with α any vector of conformal dimension, which allows (3.7)b to be written as $\| C^u K\alpha - \nu' \|_\infty \leq 1$; hence a necessary and sufficient condition for feasibility is that the infimal value of this norm over all α be less than or equal to 1. □

Remark 3.2 Infimization can be performed using Lawson's algorithm, thus for any n_u it is easy to check CSGPC feasibility at any particular t. Also, n_u need not be taken too large because the i^{th} column of Q decays to zero with increasing i, and the n_u^{th} element of $\underset{\rightarrow}{u}$, due to the input constraints, will be finite. Thus, providing that n_u is taken to be large enough so as to make the n_u^{th} column of Q sufficiently small (say 10^{-6}), the introduction of further degrees of freedom, through an increased n_u, will not affect (significantly) the solution of $Q\underset{\rightarrow}{u} = b_u$ and thus cannot affect feasibility.

3.1.2.2 Explicit conditions for *a posteriori* feasibility

Theorem 3.5 provides a procedure for investigating *a posteriori* feasibility. One

can also use Theorem 3.4 to determine *a priori* feasibility and stability: solve eqn. (3.7)a for u_t, then, still using past data, determine the range of values of u_t for which feasibility is preserved. However, in general the necessary and sufficient conditions are not explicit and thus would require the use of linear programming. It is the purpose of this section to show that in some cases the conditions are explicit and easy to derive.

Lemma 3.3 Let $a(z)$ have only one unstable pole, let that pole be at p and let $q=1/p$. Then eqns. (3.7)a,c can be rewritten as:

$$\underset{\rightarrow}{q^T \Delta u} = b_{\Delta u}; \quad b_{\Delta u} = (1-q)q^T(H_a \underset{\leftarrow}{y} - H_b \underset{\leftarrow}{u})/b(p) - u_{t-1}; \quad \text{where} \quad \underset{\rightarrow}{\Delta u} = C_\Delta \underset{\rightarrow}{u} - u_{t-1}\mathbf{e}_1 \qquad \textbf{(3.10)}$$

Proof: Using eqns. (3.10)c and (3.7)a,c we may write:

$$q^T \underset{\rightarrow}{\Delta u} = q^T(C_\Delta \underset{\rightarrow}{u} - u_{t-1}\mathbf{e}_1) = (1-q)q^T \underset{\rightarrow}{u} - u_{t-1}q^T\mathbf{e}_1 = (1-q)q^T(H_a \underset{\leftarrow}{y} - H_b \underset{\leftarrow}{u})/b(p) - u_{t-1} \qquad \textbf{(3.11)}$$

where we note that $q^TC_\Delta=(1-q)q^T$. Combining this with eqn. (3.11) yields the result□

Lemma 3.4 Let the unstable pole of Lemma 3.3 be positive, and let the system be subject to input constraints (2.28). Furthermore, let m_U and m_L denote the largest integers such that $u_{t-1}+m_UR \le U_o+U$ and $u_{t-1}-m_LR \ge U_o-U$. Then the max/min values that the left hand side of (3.10)a can assume over all feasible vectors of future values of u are given as:

$$\begin{aligned} [b^+_{\Delta u}]_{\max} &= \frac{1-q^{m_U}}{1-q}R + q^{m_U}\gamma_U; & \gamma_U &= \text{rem}[U_o+U-u_{t-1},R] \\ & & m_U &= \frac{U_o+U-u_{t-1}-\gamma_U}{R} \\ [b^+_{\Delta u}]_{\min} &= -\frac{1-q^{m_L}}{1-q}R - q^{m_L}\gamma_L; & \gamma_L &= \text{rem}[u_{t-1}-U_o+U,R] \\ & & m_L &= \frac{u_{t-1}-U_o+U-\gamma_U}{R} \end{aligned} \qquad \textbf{(3.12)}$$

where rem[a,b] is the remainder after all multiples of b have been removed from a.

Proof: All the elements of q^T are positive, hence the larger the Δu's, the larger the LHS of (3.10)a. But the Δu's are limited by R and by

$$u_{t+k} = u_{t-1} + \Delta u_t + \Delta u_{t+1} + \cdots + \Delta u_{t+k} \leq U_o + U \tag{3.13}$$

But the elements of q^T decay geometrically, hence the value of the LHS of (3.10)a is maximized when Δu_{t+i} causes u_{t+k} to reach (and maintain) its max value in minimum time (m_U+1 steps). Thus, the maximizing vector of Δu's is

$$\underset{\rightarrow}{\Delta u} = [R, R, \ldots, R, \gamma_U, 0, 0, \ldots]^T \tag{3.14}$$

where the number of repeated values of R is m_U. Substitution of this into (3.10)a yields (3.12)a. Eqn. (3.12)b can be obtained using similar arguments; the only difference is that the Δu's must be such that u reaches is lowest permissible value of U_o-U in minimum time, namely in m_L+1 time steps. The corresponding vector of control increments will have the form

$$\underset{\rightarrow}{\Delta u} = [-R, -R, \ldots, -R, -\gamma_L, 0, 0, \ldots]^T \tag{3.15}$$

□

Lemma 3.5 Let the unstable pole of Lemma 3.3 be negative and let the system be subject to input constraints (2.28). Then max/min values that the LHS of (3.10)a assumes over all feasible vectors of future u's are:

$$\begin{aligned} [\bar{b}_{\Delta u}]_{\max} &= \frac{\alpha_U + q\beta_L}{1+q} - u_{t-1}; \quad \alpha_U = \min[u_{t-1}+R,\ U_o+U];\ \ \beta_L = \max[\alpha_U - R,\ U_o - U] \\ [\bar{b}_{\Delta u}]_{\min} &= \frac{\alpha_L + q\beta_U}{1+q} - u_{t-1}; \quad \alpha_L = \max[u_{t-1}-R,\ U_o - U];\ \ \beta_U = \min[\alpha_L + R,\ U_o + U] \end{aligned} \tag{3.16}$$

Proof: This is similar to the proof above, except that the signs of the elements of q^T alternate, so the elements of the maximizing/minimizing vectors must alternate between their max and min allowable values which are defined by α_U and β_L, or β_U and α_L. □

Using the same arguments, these results can be extended to show that the maximum and minimum values the left hand side of eqn. (3.7)a can assume over all feasible values of u are given as:

$$\begin{aligned} [b_u^+]_{\max} &= \frac{u_{t-1}}{1-q} + \frac{1-q^{m_U}}{(1-q)^2}R + \frac{q^{m_U}}{1-q}\gamma_U; & [b_u^-]_{\max} &= \frac{\alpha_U + q\beta_L}{1-q^2} \\ [b_u^+]_{\min} &= \frac{u_{t-1}}{1-q} - \frac{1-q^{m_L}}{(1-q)^2}R - \frac{q^{m_L}}{1-q}\gamma_L; & [b_u^-]_{\min} &= \frac{\alpha_L + q\beta_U}{1-q^2} \end{aligned} \qquad \textbf{(3.17)}$$

Theorem 3.6 Let $a(z)$ have only one unstable pole, say at p, and let the system be subject to input constraints (2.28). Then the necessary and sufficient stability condition at t in terms of the $b_{\Delta u}$ of eqn. (3.10)b or b_u of eqn. (3.7)c is:

$$\begin{aligned} & b_{\min} \le b \le b_{\max} \qquad \text{where} \\ \text{for } p>0: \quad & b_{\min} = [b^+]_{\min}; \quad b_{\max} = [b^+]_{\max} \\ \text{for } p<0: \quad & b_{\min} = [b^-]_{\min}; \quad b_{\max} = [b^-]_{\max} \end{aligned} \qquad \textbf{(3.18)}$$

Proof: For each of the two cases considered, $b_{\min}$, $b_{\max}$ give the max/min values that the LHS of (3.10)a can assume over all feasible future control moves. Thus, if b lies in the interval $[b_{\min}, b_{\max}]$ there will exist at least one solution (or a whole family of solutions if (3.18)a holds with strict inequality) which is both feasible and does not violate the stability condition of Theorem 3.3. Conversely, by Lemmata 3.4 and 3.5 there will not exist feasible solutions which satisfy eqn. (3.10)a if b lies outside the interval $[b_{\min}, b_{\max}]$. □

Now when the condition of Theorem 3.6 is met with strict equality, the *only* feasible/stable vector of future input increments will be as proposed in the proof of Lemma 3.4 (for $p>0$) or Lemma 3.5 (for $p<0$); the former gives u a steady-state value

equal to one of the absolute constraints and the latter oscillates between α and β, never reaching a steady-state. While both cases are BIBO stable, they obviously are not desirable and do not conform to the definition of long term feasibility (LTF) as given at the beginning of the chapter. The following corollary fixes this problem by utilizing future input vectors which reach a desired steady-state value of u_∞.

Corollary 3.1 Let $a(z)$ have only one unstable pole, say at p, and let the system be subject to input constraints (2.28). Then the necessary and sufficient condition for LTF at t in terms of the $b_{\Delta u}$ of eqn. (3.10)b or b_u of eqn. (3.7)c is:

$$\begin{aligned} & b_{\min} < b < b_{\max} \qquad \text{where} \\ \text{for } p>0: \quad & b_{\min} = [b^+]_{\min}; \quad b_{\max} = [b^+]_{\max} \\ \text{for } p<0: \quad & b_{\min} = [b^-]_{\min}; \quad b_{\max} = [b^-]_{\max} \end{aligned} \tag{3.19}$$

Proof: This is similar to the proof of Theorem 3.6 except that now we must show that satisfying the condition with strict inequality allows for solutions in which the future values of u reach the desired steady-state. Consider, for example, the following vector of future input increments which causes u to reach U_o+U in min time, maintain this value, and then, at the max possible rate, reach and maintain u_∞ after n_u steps:

$$\underset{\rightarrow}{\Delta u} = [R, R, \ldots, R, \gamma_U, 0, 0, \ldots, 0, -R, -R, \ldots, -R, -\gamma_\infty, 0, 0, \ldots]^T \tag{3.20}$$

which, for positive p, makes the value of the LHS of (3.10)a,

$$\begin{aligned} & \underset{\rightarrow}{q^T \Delta u} = \frac{1-q^{m_U}}{1-q} R + q^{m_U} \gamma_U - q^{n_u - m_\infty} \left(\frac{1-q^{m_\infty}}{1-q} R + q^{m_\infty} \gamma_\infty \right) \\ & \gamma_U = \text{rem}[U_o + U - u_{t-1}, R]; \qquad \gamma_\infty = \text{rem}[U_o + U - u_\infty, R] \\ & m_U = \frac{U_o + U - u_{t-1} - \gamma_U}{R}; \qquad m_\infty = \frac{U_o + U - u_\infty - \gamma_\infty}{R} \end{aligned} \tag{3.21}$$

This differs from $[b_{\Delta u}{}^+]_{\max}$ by a factor which is multiplied by $q^{n_u - m_\infty}$ where n_u can be

chosen arbitrarily large so as to make this factor arbitrarily small, thus making the value of the LHS of (3.10)a arbitrarily close to $[b_{\Delta u}^{+}]_{max}$. Similarly, it can be shown that there exist vectors which give u a desired steady-state value and make the LHS of eqn. (3.10)a or eqn. (3.7)a arbitrarily close to any of the limits of the interval $[b_{min}, b_{max}]$. □

These same ideas carry over to the case where $a(z)$ has two unstable real poles, for which the matrix Q of eqn. (3.10) will have two row vectors, $\boldsymbol{q}^T_1$, $\boldsymbol{q}^T_2$, and the right hand side of (3.10) will be a two dimensional vector, say $\boldsymbol{b}_{\Delta u}=[b_1,b_2]^T$. However, applying Theorem 3.6 to $\boldsymbol{q}_1^T\underset{\rightarrow}{\Delta \boldsymbol{u}}=b_1$ and $\boldsymbol{q}_2^T\underset{\rightarrow}{\Delta \boldsymbol{u}}=b_2$ independently will only generate necessary conditions; below we state conditions which are both necessary and sufficient.

Theorem 3.7 Let $a(z)$ have 2 unstable real poles at p_1, p_2. Then the matrix Q of (3.10)a will have 2 rows, $\boldsymbol{q}^T_1$, $\boldsymbol{q}^T_2$, for $q_1=1/p_1$ and $q_2=1/p_2$. Let $[b_1]_{max}$ and $[b_1]_{min}$ be as per Theorem 3.6, and let $\Delta \boldsymbol{u}_1$ be the vector of future control increments for which $\boldsymbol{q}^T_1\Delta \boldsymbol{u}_1$ attains its max value of $[b_1]_{max}$. Then the necessary and sufficient LTF conditions are:

$$[b_1]_{min} < b_1 < [b_1]_{max}; \quad \boldsymbol{q}_2^T\Delta \boldsymbol{u}_1 - \max_{x}\{\boldsymbol{q}_2^T\boldsymbol{x}\} < b_2 < \boldsymbol{q}_2^T\Delta \boldsymbol{u}_1 - \min_{x}\{\boldsymbol{q}_2^T\boldsymbol{x}\} \quad \textbf{(3.22)}$$

where the vector $\boldsymbol{x}$ is constrained to satisfy the condition:

$$\boldsymbol{q}_1^T\boldsymbol{x} = [b_1]_{max} - b_1 \quad \textbf{(3.23)}$$

Proof: By Corollary 3.1, condition (3.22)a guarantees that a $\underset{\rightarrow}{\Delta \boldsymbol{u}}$ exists which satisfies $\boldsymbol{q}_1^T\underset{\rightarrow}{\Delta \boldsymbol{u}}=b_1$ and leaves u at the desired steady-state value. The totality of such vectors can be written as $\underset{\rightarrow}{\Delta \boldsymbol{u}}=\Delta \boldsymbol{u}_1-\boldsymbol{x}$, where $\boldsymbol{x}$ must satisfy eqn. (3.23). Such $\underset{\rightarrow}{\Delta \boldsymbol{u}}$ will make $\boldsymbol{q}_2^T\underset{\rightarrow}{\Delta \boldsymbol{u}}$ equal to $\boldsymbol{q}_2^T\Delta \boldsymbol{u}_1-\boldsymbol{q}_2^T\boldsymbol{x}$, which is maximized and minimized by the interval defined in condition (3.22)b. Clearly then, the equation $Q\underset{\rightarrow}{\Delta \boldsymbol{u}}=\boldsymbol{b}_{\Delta u}$ will admit a feasible solution

(with u going to the desired steady-state value) if, and only if, both conditions (3.22)a and (3.22)b are satisfied. □

In order to invoke Theorem 3.7, one needs to determine the maximizing/minimizing vectors $\boldsymbol{x}$, but this is straightforward when p_1, and p_2 are real and share the same sign. Then it can be shown that the $\boldsymbol{x}$ that maximizes $\boldsymbol{q}^T_2\boldsymbol{x}$ is the vector causing the vector of future u's corresponding to $\underset{\rightarrow}{\Delta \boldsymbol{u}}=\Delta \boldsymbol{u}_1-\boldsymbol{x}$, to move toward U_o-U for as long as possible (as dictated by eqn. (3.23) and U_o-U), and then, at the maximum possible rate, reach U_o+U. The detailed proof for the structure of $\boldsymbol{x}$ is simple but long and will be omitted. Here we simply make two obvious remarks: (a) since the starting point, u_t, is common to both $\boldsymbol{u}_1$ and $\underset{\rightarrow}{\boldsymbol{u}}$, and since the ultimate value of U_o+U is also common to both, the sum of the elements of the vector $\boldsymbol{x}$ must be zero; (b) because the elements of $\boldsymbol{q}^T_2$ decay as a geometric series, the vector which maximizes $\boldsymbol{q}^T_2\boldsymbol{x}$ must have as large a front end as eqn. (3.23) and the absolute and rate constraints will permit. A similar procedure can be used to determine the minimizing $\boldsymbol{x}$.

The case of two real unstable poles of different sign, or the case of two complex conjugate unstable poles is considerably more complicated and will not be given here. In cases like this as well as for the general case of any number of unstable poles, one must revert to Theorem 3.5 for a necessary and sufficient test of feasibility.

3.1.2.3 Illustrative examples

Here, we illustrate the instability of constrained predictive control algorithms when they encounter short term infeasibility with an open-loop unstable system. The

algorithm we use for this is CSGPC, but the property is true for all.

Example 3.1 Let the system with $a(z)=1-2.2z^{-1}+0.09z^{-2}+0.252z^{-3}$, $b(z)=2+0.45z^{-1}+z^{-2}$ be subject to the input constraints with $U_o=0$, $U=25$, $R=0.04$; $a(z)$ has only one unstable pole (at 2.1), so stability can be tested with the explicit conditions of Theorem 3.6. Assume the system is at rest, $r=0$ for $t\leq 1$ and $r=1$ for $t>1$. Of course, $u_t=0$ for $t\leq 1$, and at $t=2$ $m_U=m_L=U/R=625$; $m_U=m_L$ because the absolute limits are symmetric about $u_1=0$. Since R is small, m_U and m_L will remain very large for all values of t during the simulation, and so q^{m_U} and q^{m_L} will be insignificantly small, (because $q=1/2.1$). Then, from eqn. (3.12), we have:

$$b_{\max}=[b^{+}_{\Delta u}]_{\max}=\frac{R}{1-q}=0.0764; \quad \text{and} \quad b_{\min}=[b^{+}_{\Delta u}]_{\min}=-\frac{R}{1-q}=-0.0764 \qquad \textbf{(3.24)}$$

The necessary and sufficient stability condition, therefore, is that for $t=2,3,\ldots,9$ the $b_{\Delta u}$ of eqn. (3.10)b must lie between -0.0764 and 0.0764. Upon application of CSGPC, the optimal value of Δu_2, as seen from Figure 3.1c, is 0.04, but for this the corresponding value of $b_{\Delta u}$, as shown in Figure 3.1d, is -0.084. Thus, the first control move recommended by CSGPC results in instability, so that at $t=3$ there will not exist a $\vec{\Delta u}$ for which conditions (2.28)b and (3.7)a,b can be satisfied simultaneously. However, the input constraints are hard, so eqn. (2.28)b will be satisfied; as a consequence, condition (3.7)a will be violated and the feedback system will go unstable. This is illustrated in Figure 3.1a which shows the response of the output. Infeasibility means lack of feasibility over an infinite horizon which also implies short term infeasibility; as a result the infimal value of $\|Ac-v(t)\|_\infty$ will be greater than 1. Furthermore, due to instability this infimal value diverges (Figure 3.1b). If the unstable pole were at 2 instead of 2.1

the feasibility interval would become [-0.084,0.084] and so CSGPC would operate at the limit of BIBO stability as shown in Figure 3.2.

Example 3.2 In the example above, short term infeasibility caused CSGPC to choose control moves which render the problem unstable. This need not always be the case, especially if the short term infeasibility concerns future constraint violations. Through the use of the MWLS cost, CSGPC will choose c so as to minimize the worst case constraint violations, and if these violations are in the future, then CSGPC will be able to reduce this violation further at the next step. By the time we come to implement the offending value of c, infeasibility may have disappeared altogether.

The system to be considered has a transfer function with $a(z)=1-1.3z^{-1}+0.144z^{-3}$, $b(z)=2+0.45z^{-1}+z^{-2}$ and has only one unstable pole (at 1.2). The system input is subject to constraints defined by $U_o=0$, $U=0.05$ and $R=0.2$. The values of $b_{\min}$ and $b_{\max}$ are calculated to be -0.3 and 0.3 respectively, and the corresponding value of b_u is plotted in Figure 3.3d; clearly, CSGPC satisfies the necessary and sufficient stability condition, despite the fact that it runs up against short term infeasibility as demonstrated by Figure 3.3b. CSGPC manages to recover short term feasibility a few time steps later; as a result the algorithm gives a stable and satisfactory output response (Figure 3.3a).

Because these results imply a test which is *a posteriori*, they really only serve to show how close a system is coming to risking instability, or show why instability occurred. These conditions must be propagated into the future to provide a useful limit on the choice of future control moves; this is done below.

3.1.3 *A priori* conditions for the existence of a stabilizing solution

In this section, we present the necessary and sufficient *a priori* conditions for which a stabilizing solution will continue to exist. The conditions of the previous section can be propagated into the future for systems with absolute and rate input constraints, but the result, though straightforward, is quite cumbersome. For clarity of presentation, we consider systems subject only to absolute or rate input constraints, in which case the bounds on the left hand sides of eqns. (3.7) or (3.10), $b_{\min}$ and $b_{\max}$, are time invariant.

Lemma 3.6 Let $a(z)$ have only one unstable pole at $p=1/q$, and let the system be subject only to absolute input constraint (2.28)a. Then, the bounds on the LHS of eqn. (3.7) are:

$$[b_u]_{\max} = \frac{U_o}{1-q} + \frac{U}{1-|q|}; \qquad [b_u]_{\min} = \frac{U_o}{1-q} - \frac{U}{1-|q|} \tag{3.25}$$

Proof: Without rate constraints, $R=\infty$, so, for $p>1$, $\gamma_U=U_o+U-u_{t-1}$, $\gamma_L=u_{t-1}-U_o+U$, and $m_U=m_L=0$. Therefore, from eqns. (3.17)a,b, $[b_u^+]_{\max}=(U_o+U)/(1-q)$ and $[b_u^+]_{\min}=(U_o-U)/(1-q)$. For $p<-1$, $\alpha_U=\beta_U=U_o+U$ and $\alpha_L=\beta_L=U_o-U$. Therefore, from eqns. (3.17)c,d, $[b_u^-]_{\max}=U_o/(1-q)+U/(1+q)$ and $[b_u^-]_{\min}=U_o/(1-q)-U/(1+q)$. □

Lemma 3.7 Let $a(z)$ have only one unstable pole at $p=1/q$, and let the system be subject only to rate input constraint (2.28)b. The bounds on the LHS of eqn. (3.10) are:

$$[b_{\Delta u}]_{\max} = \frac{R}{1-|q|}; \qquad [b_{\Delta u}]_{\min} = \frac{-R}{1-|q|} \tag{3.26}$$

Proof: Without absolute constraints, $U=\infty$, so, for $p>1$, $m_U=m_L=\infty$, and therefore, from eqn. (3.12), $[b_{\Delta u}^+]_{\max}=R/(1-q)$ and $[b_{\Delta u}^+]_{\min}=-R/(1-q)$. For $p<-1$, $\alpha_U=u_{t-1}+R$, $\alpha_L=u_{t-1}-R$, and $\beta_U=\beta_L=u_{t-1}$. Therefore, from eqn. (3.16), $[b_{\Delta u}^-]_{\max}=R/(1+q)$ and $[b_{\Delta u}^-]_{\min}=-R/(1+q)$. □

These time invariant bounds make the projection of the *a posteriori* conditions of the previous section into the future trivial. An input which violates the resulting *a priori* bounds will lead to instability.

Theorem 3.8 Let $a(z)$ have only one unstable pole at $p=1/q$, and let the system be subject only to input absolute constraint (2.28)a. Then, at time t, a stabilizing solution will be guaranteed to exist at the next time instant (ie. the problem will be LTF) if, and only if, u_t is chosen such that:

$$b_u - \frac{qU_o}{1-q} - \frac{|q|U}{1-|q|} < u_t < b_u - \frac{qU_o}{1-q} + \frac{|q|U}{1-|q|} \tag{3.27}$$

Proof: Extracting u_t from eqn. (3.7)a (written for one unstable pole) gives:

$$u_t = b_u - q\mathbf{q}^T \underset{\rightarrow t+1}{\mathbf{u}} \tag{3.28}$$

The result follows from application of the time invariant bounds of Lemma 3.6 and from arguments given in the proofs of Theorem 3.6 and Corollary 3.1. □

Theorem 3.9 Let $a(z)$ have only one unstable pole at $p=1/q$, and let the system be subject only to input rate constraint (2.28)b. Then, at time t, LTF will be maintained at the following time instant if, and only if, Δu_t is chosen such that:

$$b_{\Delta u} - \frac{|q|R}{1-|q|} < \Delta u_t < b_{\Delta u} + \frac{|q|R}{1-|q|} \tag{3.29}$$

Proof: Extracting Δu_t from eqn. (3.10)a gives:

$$\Delta u_t = b_{\Delta u} - q\mathbf{q}^T \underset{\rightarrow t+1}{\Delta \mathbf{u}} \tag{3.30}$$

The result follows from application of the time invariant bounds of Lemma 3.7 and from arguments given in the proofs of Theorem 3.6 and Corollary 3.1. □

The conditions given above provide limits which the current input must not violate; this will be illustrated by example in Chapter 6 after the inclusion of disturbances.

3.2 Stability conditions for the general case

Corollary 3.1 and Theorems 3.7-9 provide explicit conditions for LTF for systems with only one or two real unstable poles; for the general case, Theorem 3.3 gives necessary and sufficient stability conditions, but assumes that n_u is arbitrarily large. As per Remark 3.1, in practice n_u is taken to be finite, and this leads to the implementation of Theorem 3.4. From a computational viewpoint, it is desirable/essential to keep n_u small, which leads to sufficient only results; this is because u is a finite length sequence (FLS). The future u FLS of Section 3.1 must: (i) cancel the unstable effects of the open-loop polynomial (as per constraint (3.7)a); and (ii) respect the input constraints (2.28). In practice these requirements may necessitate the use of a large number of degrees of freedom (ie. a large n_u). That the future u prediction is a FLS implies that it has no poles, which means all system zeros, $b(z)$ (poles in the inverse plant), must have been "cancelled". The previous section showed that the effect of $\boldsymbol{q}^T$ is to "cancel" the *unstable* poles of $a(z)$; this is the right idea, but can of course be done more directly. To remedy these problems, here we develop stability and asymptotic tracking conditions which do not depend on the use of FLSs. The key lies in "cancelling" only those roots which must be "cancelled" (ie. those outside the unit circle), both in the output predictions and in the input predictions; the result is predictions which are infinite length sequences (ILS). As in the previous section, we will require not only BIBO stability, but also asymptotic tracking in the predictions.

3.2.1 Stability conditions using ILS inputs

Eqn. (2.9) can be written in a z-transform equivalent form as:

$$\alpha(z)y(z)=b(z)\Delta u(z)+p(z) \qquad \text{or} \qquad y(z)=\frac{b(z)\Delta u(z)+p(z)}{a^{-}(z)\alpha^{+}(z)} \tag{3.31}$$

where the coefficients of the powers of z^{-1} in $y(z)$, $\Delta u(z)$, $p(z)$ are the elements of $\underset{\rightarrow}{\Delta u}$, $\underset{\rightarrow}{y}$, and $\underset{\rightarrow}{v_p}=H_b\underset{\leftarrow}{\Delta u}-H_\alpha\underset{\leftarrow}{y}$, taken in order; $a^{-}(z)$ and $\alpha^{+}(z)=\Delta(z)a^{+}(z)$ are the factors of $\alpha(z)$, with order n_{a^-} and $n_{a^+}+1$, whose roots lie inside and outside (or on) the unit circle, respectively. It is apparent that the necessary and sufficient condition for the stability of the predicted output, y, is that the numerator of eqn. (3.31)b contain as a factor (and thus cancel) the unstable system poles, namely that:

$$b(z)\Delta u(z)+p(z)=\alpha^{+}(z)\phi(z) \qquad \text{or} \qquad \Delta u(z)=\frac{\alpha^{+}(z)\phi(z)-p(z)}{b^{-}(z)b^{+}(z)} \tag{3.32}$$

where $\phi(z)$ is the z-transform of a convergent sequence $\{\phi_0,\phi_1,\phi_2,\ldots\}$ and $b^{-}(z)$, $b^{+}(z)$ are defined in a manner analogous to $a^{-}(z)$, $a^{+}(z)$. It follows that the predicted trajectories of control increments, Δu, will be stable if, and only if, the numerator of eqn. (3.32)b contains the "unstable" system zeros:

$$\alpha^{+}(z)\phi(z)-p(z)=b^{+}(z)\psi(z) \qquad \text{or} \qquad \alpha^{+}(z)\phi(z)-b^{+}(z)\psi(z)=p(z) \tag{3.33}$$

where $\psi(z)$ is a polynomial with a convergent sequence of coefficients. Eqn. (3.33)b constitutes an equality constraint which must be satisfied by the otherwise arbitrary $\phi(z)$ and $\psi(z)$; a particular minimal order solution is given by the vector of coefficients, $[\phi_p^T,\psi_p^T]=[\phi_0,\phi_1,\ldots,\phi_{n_\phi},\psi_0,\psi_1,\ldots,\psi_{n_\psi}]^T$ of the z-transforms, $\phi_p(z),\psi_p(z)$, $n_\phi=\max[n_b,n_a]-1$, $n_\psi=n_{a^+}$:

$$\left[\Gamma_{\alpha^+} \;\; -\Gamma_{b^+}\right]\begin{bmatrix}\phi_p\\ \psi_p\end{bmatrix}=\begin{bmatrix}v_p\\ \mathbf{0}\end{bmatrix} \tag{3.34}$$

where Γ_{α^+}, Γ_{b^+} are matrices containing the first $n_\phi+1$, $n_\psi+1$ columns of the

$\{n_\phi+n_\psi+2\}\times\{n_\phi+n_\psi+2\}$ convolution matrices C_{α^+}, C_{b^+}; the degree of $p(z)$ is n_a; and $\mathbf{0}$ is a vector of zeros of conformal dimension. With $\alpha^+(z)$ and $b^+(z)$ coprime, the particular solution is:

$$\begin{bmatrix}\phi_p\\ \psi_p\end{bmatrix}=\begin{bmatrix}\Gamma_{\alpha^+} & -\Gamma_{b^+}\end{bmatrix}^{-1}\begin{bmatrix}v_p\\ \mathbf{0}\end{bmatrix}=\begin{bmatrix}P_1 & P_3\\ P_2 & P_4\end{bmatrix}\begin{bmatrix}v_p\\ \mathbf{0}\end{bmatrix}=\begin{bmatrix}P_1\\ P_2\end{bmatrix}v_p \tag{3.35}$$

where $P_{1\text{-}4}$ denote partitions of the inverse in (3.35) of dimensions conformal to ϕ_p, ψ_p, v_p, $\mathbf{0}$. Taking z-transforms, it is easy to show that the general solution to (3.33)b is:

$$\phi(z)=b^+(z)c(z)+\phi_p(z);\qquad \psi(z)=\alpha^+(z)c(z)+\psi_p(z) \tag{3.36}$$

where $c(z)$ is a polynomial with a convergent sequence of coefficients and contains the available degrees of freedom. Inserting (3.36)a,(3.36)b into (3.32)a,(3.33)a and then into (3.31)b,(3.32)b gives:

$$y(z)=\frac{b^+(z)c(z)+\phi_p(z)}{a^-(z)};\qquad \Delta u(z)=\frac{\alpha^+(z)c(z)+\psi_p(z)}{b^-(z)} \tag{3.37}$$

It then follows that for asymptotic tracking the coefficients c_i must have a limit, say c_∞:

$$c_\infty=\frac{a^-(1)}{b^+(1)}\lim_{z\to1}[1-z^{-1}]\,y(z)=\frac{a^-(1)}{b^+(1)}r \tag{3.38}$$

where we note that the sequence ϕ_p is of finite length, $n_\phi+1$, and thus has zero steady-state value.

For practical implementation, the number of degrees of freedom used for optimization has to be finite (and preferably small); thus, as was done with the sequence of future inputs in Section 3.1, here we require $c(z)$ to be a FLS and to reach its steady-state value within n_c+1 steps; n_c shall be termed the command horizon. We emphasize that, despite the FLS nature of c, in general, the sequence of future control increments will be ILS (as implied by eqn. (3.37)b), and thus it will avoid the conflict that may exist between FLSs and input constraints. Other predictive control formulations (eg [30], [54],

[18], [24]) use inputs or input increments as degrees of freedom and thus implicitly constrain them to be FLS; here we impose no such restriction.

With $c(z)=c_0+c_1z^{-1}+\cdots+c_{n_c-1}z^{-n_c+1}+c_\infty z^{-n_c}/(1-z^{-1})$, eqns. (3.37) can be written as:

$$\begin{aligned} \underset{\rightarrow}{y} &= \Gamma_{b^+/a^-}\,\underset{\rightarrow}{c} + \theta_{b^+/a^-}\,c_\infty + \Gamma_{1/a^-}P_1\underset{\rightarrow}{v}_p \\ \Delta\underset{\rightarrow}{u} &= \Gamma_{\alpha^+/b^-}\,\underset{\rightarrow}{c} + \theta_{\alpha^+/b^-}\,c_\infty + \Gamma_{1/b^-}P_2\underset{\rightarrow}{v}_p \\ \underset{\rightarrow}{u} &= \Gamma_{a^+/b^-}\,\underset{\rightarrow}{c} + \theta_{a^+/b^-}\,c_\infty + \Gamma_{1/\Delta b^-}P_2\underset{\rightarrow}{v}_p + u_{t-1}\mathbf{1} \end{aligned} \qquad \textbf{(3.39)}$$

where the i^{th} column of the Γ matrices contains the impulse response of the indicated transfer function multiplied by z^{-i+1}, and the θ vectors contain the step response of the indicated transfer function multiplied by z^{-n_c}. Eqn. (3.39)c is derived from i) eqn. (3.39)b, ii) $\underset{\rightarrow}{u}=C_\Delta^{-1}\Delta\underset{\rightarrow}{u}+u_{t-1}\mathbf{1}$, and iii) $a^+(z)=\alpha^+(z)/\Delta(z)$. C_Δ^{-1} is a square lower triangular matrix of 1's, whereas the elements of the vector $\mathbf{1}$ are all 1.

To get the *a priori* stability condition, we need to determine the allowable interval for u_t, the first element of $\underset{\rightarrow}{u}$ in prediction eqn. (3.39)c, such that eqns. (3.39)b,c satisfy input constraints (2.28). Thus, we must minimize/maximize over $\underset{\rightarrow}{c}$ the first element of eqn. (3.39)c, $u_t=e_1^T\underset{\rightarrow}{u}$, subject to (2.28). Note that the part of $u_t=e_1^T\underset{\rightarrow}{u}$ which is not a function of $\underset{\rightarrow}{c}$ is constant and will be ignored in the cost (but not the constraints) of the optimization:

$$\min_{\underset{\rightarrow}{c}} \; \pm e_1^T\,\Gamma_{a^+/b^-}\,\underset{\rightarrow}{c} \qquad s.t. \quad \begin{cases} u_L \le \underset{\rightarrow}{u} \le u_U \\ r_L \le \Delta\underset{\rightarrow}{u} \le r_U \end{cases} \qquad \textbf{(3.40)}$$

where the dependence of $\underset{\rightarrow}{u}$, $\Delta\underset{\rightarrow}{u}$ on $\underset{\rightarrow}{c}$ is as defined in eqns. (3.39)b,c. By eqn. (3.39)c, the solutions of optimization (3.40), $\underset{\rightarrow}{c}_{\max}$, $\underset{\rightarrow}{c}_{\min}$, give vectors, $\underset{\rightarrow}{u}_{\max}$, $\underset{\rightarrow}{u}_{\min}$, the first elements of which define the allowable interval for u_t as:

$$[u_t]_{\min} \leq u_t \leq [u_t]_{\max} \qquad \text{where} \qquad u_t = \mathbf{e}_1^T \underset{\rightarrow}{\boldsymbol{u}} \tag{3.41}$$

Theorem 3.10 The necessary and sufficient *a priori* stability condition is that u_t must lie in the interval defined by eqn. (3.41) where $[u_t]_{\min}$ and $[u_t]_{\max}$ are the minimum and maximum values of u_t, as computed by optimization (3.40).

Proof: By derivation, the totality of stable input vectors $\underset{\rightarrow}{\boldsymbol{u}}$ which give stable output vectors must be of the form of eqn. (3.39c); since we are dealing with hard constraints, failure to meet (3.40b) implies violation of (3.39) and will lead to instability.

Constraints (3.40b) are linear, so the set of $\underset{\rightarrow}{\boldsymbol{c}}$ that satisfy (3.40b) is convex and the vector $\boldsymbol{c}_\epsilon = \boldsymbol{c}_{\min} + \epsilon(\boldsymbol{c}_{\max} - \boldsymbol{c}_{\min})$ will also satisfy (3.40b) for $0 \leq \epsilon \leq 1$. However, as ϵ varies continuously from 0 to 1, $u_t = \mathbf{e}_1^T \underset{\rightarrow}{\boldsymbol{u}}$ also varies continuously between $[u_t]_{\min}$ and $[u_t]_{\max}$. Thus, to every value in this interval there will correspond at least one $\underset{\rightarrow}{\boldsymbol{c}}$ such that the vector $\underset{\rightarrow}{\boldsymbol{u}}$ satisfies constraints (3.40b) and, by definition, gives stable inputs/outputs; namely, providing that stability interval (3.41) is satisfied, there exists at least one stabilizing, feasible $\underset{\rightarrow}{\boldsymbol{u}}$. □

Remark 3.3 Strictly speaking the necessity of Theorem 3.10 holds for the case when the command horizon n_c is taken to be arbitrarily large, thus at first sight, this result appears to be of limited practical use. However, in practice n_c must be taken to be finite (and preferably small), and this will limit the stability interval; the condition of Theorem 3.10 is then necessary and sufficient *for a given* command horizon n_c. Thus, if optimization (3.40) is infeasible for a given n_c, then there does not exist a stabilizing control trajectory of the form of (3.39) and n_c **must** be increased; this is in contrast to the previous section,

where infeasibility need not be intrinsically due to the use of a short horizon, but rather could be attributed to the FLS property which is arbitrarily imposed on u.

The ILS nature of the predicted u used in this section implies (at first sight) that constraint violations need to be checked over an infinite horizon (regardless of the length of the command horizon). However, because the transfer function from the command input, c, to the system input, u, is stable, bounding techniques like those given in [30], [54], [41] can be employed by which constraint satisfaction over some finite input horizon implies satisfaction over the infinite horizon. This will be shown in Chapter 5.

3.2.2 Illustrative example

An obvious use of these stability results is in a supervisory capacity when using control strategies which either do not have a stability guarantee altogether or lose the guarantee temporarily due to short term infeasibility (STIF) (eg. due to large set-point changes). When the controller asks for inputs inside the stability interval, no remedial action is required, but if the input is outside the interval, the input could be clipped or control of the system could be switched to a known stable controller.

It is pointed out that *all* terminally constrained predictive algorithms require short term feasibility (STF) and thus lack stability guarantees for horizons which are too short to avoid STIF. In practice, short horizons are essential in that they imply short computing times; for systems with relatively fast dynamics, the luxury of large optimizations may not be possible. Here we show how the stability conditions are used to guarantee the stable operation of predictive algorithms when horizons are too short to ensure STF (at all times). The predictive algorithm we use is CSGPC. In common with

terminally constrained predictive algorithms, the optimizations of Theorem 3.10 (required to find the stability interval) also requires feasibility, but through the use of ILS's in place of FLS's, it is doing only what is necessary and thus will usually be feasible with much shorter horizons.

This is demonstrated in the example where the CSGPC optimization becomes STIF due to a large set-point change. When CSGPC is STIF, one needs a logical procedure for choosing u_t within the *a priori* stability interval of eqn. (3.41). One such strategy is to choose u_t within the limits such that the maximum future constraint violations *predicted* by CSGPC are minimized. Since CSGPC uses predicted input sequences which are FLSs, it is easy to formulate this minimization as a linear program (LP); in this we can introduce weighting, w_j, $w_{j-1} \geq w_j$, to place more emphasis on the early predicted constraint violations:

$$\begin{aligned} &\min \max[w_j(U_o - U - u_{t+j}), w_j(u_{t+j} - U_o - U), w_j(-R - \Delta u_{t+j}), w_j(\Delta u_{t+j} - R)];\ 0 < j < n_u \\ &\quad s.t. \quad y_{t+n_y+i+1} = r; \quad \Delta u_{t+n_u+i} = 0; \quad \forall\ i \geq 0; \qquad [u_t]_{\min} \leq u_t \leq [u_t]_{\max} \end{aligned} \tag{3.42}$$

The rationale (of minimizing worst case predicted constraint violations) is as follows: of all the predicted u's, only u_t is actually implemented, therefore the choice of u_t (within the *a priori* stability interval) can be governed by minimizing future *predicted* constraint violations and thus maximizing the possibility of a return to STF at the next time instant.

Algorithm 3.1 At each time t:

Step 1 Test the CSGPC optimization for STF

Step 2 If STF, apply CSGPC; implement u_t, increment t, and return to step 1

Step 3 If STIF, calculate the stability interval for u_t from optimization (3.40) ($[u_t]_{\min}$, $[u_t]_{\max}$) and implement the value of u_t which optimizes eqn. (3.42); increment t and return

to step 1

CSGPC requires, at most, two optimizations: an LP to find a feasible initial solution (which, at the same time, tests for STF) and a quadratic program (QP) to minimize the predicted cost function. Algorithm 3.1 (when CSGPC is STIF) requires up to four LP's: LP1 to test CSGPC for STF, LP2,3 to determine the upper and lower limits on u_t, and LP4 to minimize the future predicted CSGPC constraint violations. Ignoring the difference in computing times of QP's versus LP's (which would amplify our point), we will show that, through the use of smaller optimizations, Algorithm 3.1 will be significantly faster than CSGPC, when the latter is implemented with horizons long enough to ensure STF at all times. We will compare the theoretical maximum number of iterations, the size of the optimizations, and the maximum computation time spent doing optimizations at any given time step. The optimization routines used will be LP and QP from Matlab's Optimization Toolbox, which employ Bland's anti-cycling rule. As a consequence, the theoretical maximum number of iterations is bounded by the number of different subsets of constraints with k elements, where k varies from zero to the number of DOF; in practice, computation time tends to vary with the size of the optimization [3]. This size is $n_c \times m$, the product of the number of DOF and the total number of constraints.

Example 3.3 Let the system transfer function $g(z)=z^{-1}b(z)/a(z)$ be defined by

$$\begin{aligned} a(z) &= (1-0.6z^{-1})(1-2z^{-1})(1+3z^{-1})(1-4z^{-1}+2.25z^{-2}) \\ b(z) &= (1-0.2z^{-1})(1-0.8z^{-1})(1-0.6z^{-1}+0.25z^{-2}) \end{aligned} \tag{3.43}$$

let the constraints be $U_o=0$, $U=60$, $R=10$, assume zero initial conditions, and apply a

unit step set-point. Figure 3.4 shows the results for CSGPC with n_u=14 and n_y=12 (8 DOF); with fewer DOF, CSGPC runs into STIF and goes unstable. In this figure and the following, the left plot depicts output responses (in solid lines), and set-point trajectories (dashed lines), whereas the right plot depicts control (solid lines) and control increment (dash-dotted lines, scaled by a factor of 6) trajectories. For CSGPC with 8 DOF, the LP and QP sizes are 9×54 and 8×54, totalling 918; the theoretical maximum number of iterations are 1.97×10^{15} and 4.29×10^{13}. The longest computation time required by the optimizations at any sampling time was 1.15 seconds. Figure 3.5 shows the results for Algorithm 3.1 with n_c=2. With this command horizon, LP1 and LP4 are 3×30 and LP2,3 are 2×116, totalling 644 (vs. 918 for CSGPC); also the theoretical maximum number of iterations are much more reasonable: 25261 (for LP1,4) and 13457 (for LP2,3). The longest computation time required at any sampling instant was 0.88 seconds; an improvement of almost 25%. It is interesting to note that despite the very small number of DOF used by Algorithm 3.1 (2 vs. 8 used by CSGPC), the results of Figure 3.5 are of comparable quality (by way of maximum overshoot, speed of response, level of control activity, etc) to those of Figure 3.4.

3.3 Chapter summary

The results of this chapter enable the determination of necessary and sufficient *a priori* stability conditions and incorporate conditions for asymptotic tracking. Two practical schemes are considered, one which gives explicit stability conditions for systems with one unstable pole and the other which gives stability conditions for the general case through the use of linear programs. By allowing the predicted u to be an ILS, the latter leads to

an optimization problem which remains feasible for smaller horizons, thereby enabling a significant reduction in computational load, while retaining the guarantee of stability; this point is clearly illustrated by means of the final numerical example.

In the next chapter, we take an alternate approach to the problem of short term feasibility. Chapter 4 offers several modifications to CSGPC which serve to avoid STIF during set-point changes and therefore, to avoid instability.

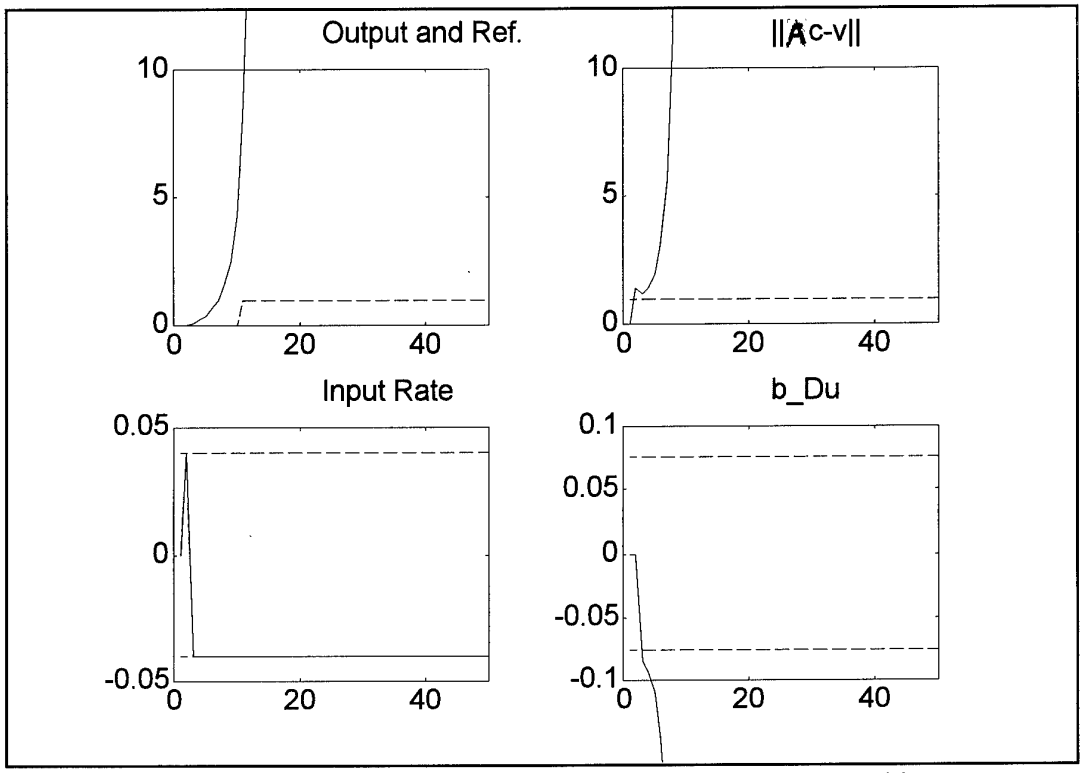


Figure 3.1 Example 3.1 - CSGPC with short term infeasibility - unstable

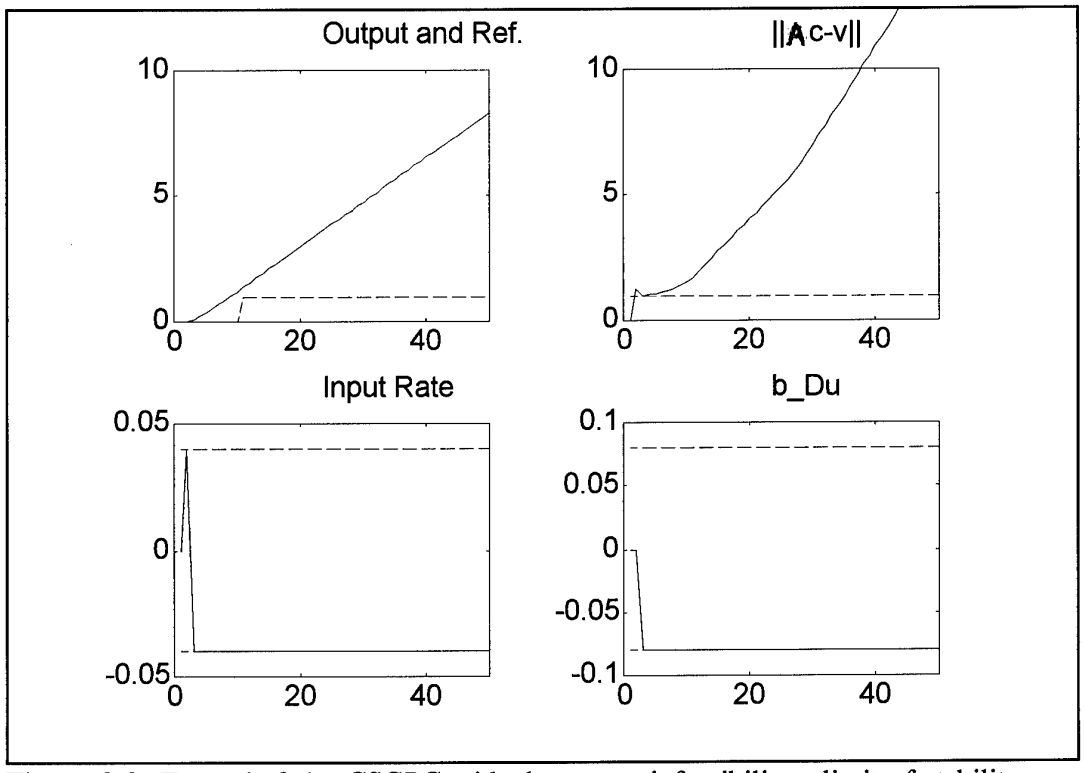


Figure 3.2 Example 3.1 - CSGPC with short term infeasibility - limit of stability

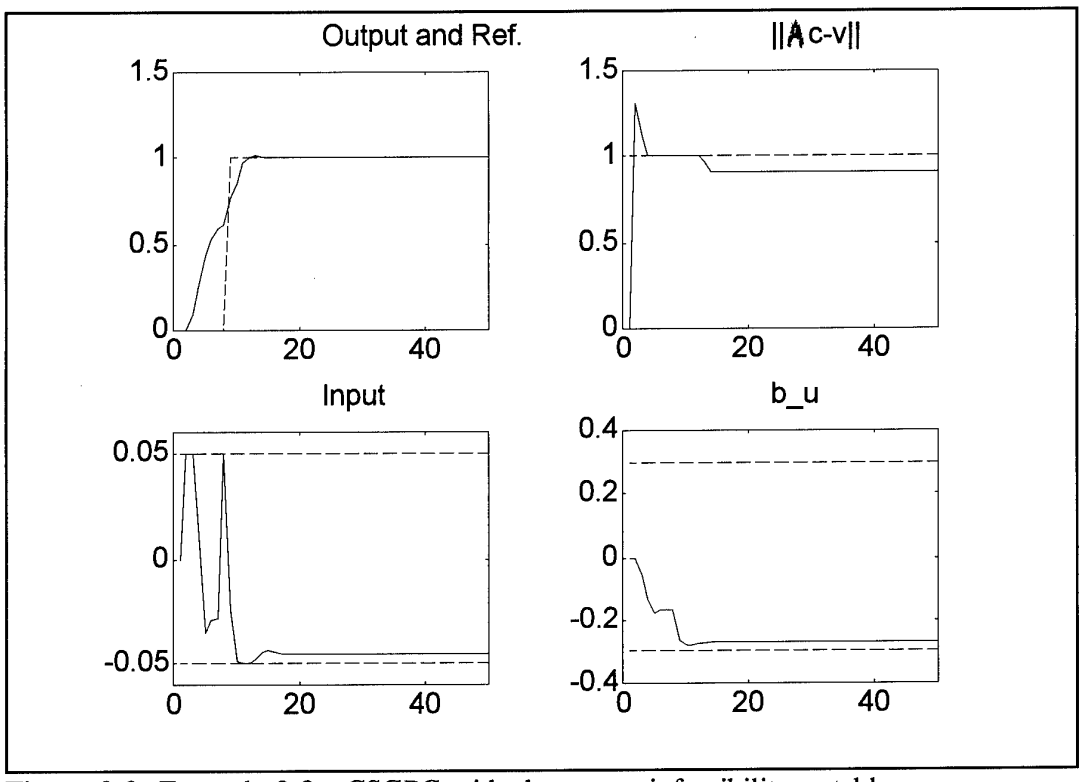


Figure 3.3 Example 3.2 - CSGPC with short term infeasibility - stable

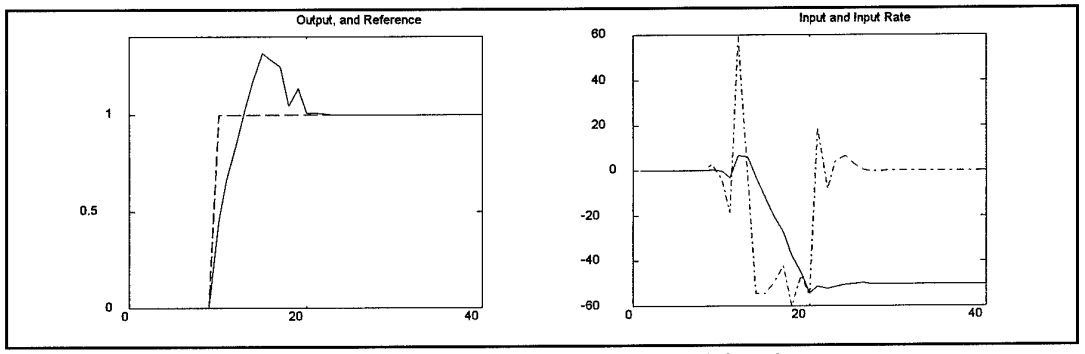


Figure 3.4 Example 3.3 - CSGPC with eight degrees of freedom

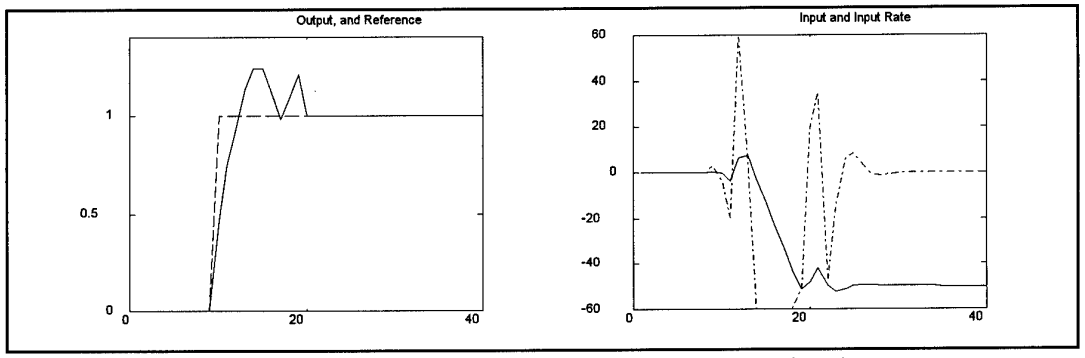


Figure 3.5 Example 3.3 - Algorithm 3.1 with two degrees of freedom

Chapter 4

Modifications to Constrained Stable Generalized Predictive Control

Theorem 2.2 states that if CSGPC is feasible for a finite n_y (ie. short term feasible, or STF), then the algorithm will lead to asymptotic stability. However, nothing can be said about the stability of CSGPC in the case of short term infeasibility (STIF). As was seen in Example 3.2, by minimizing the worst case future constraint violation, CSGPC can overcome STIF and give stability and asymptotic tracking. This property is extremely useful, but cannot be guaranteed in general. A common cause of STIF is large set-point changes; in this chapter, we propose three different modifications to CSGPC which all retain feasibility during large set-point changes and thus lead to stability.

STIF is caused by the requirement that the predicted output should reach its new target value within n_y steps (a terminal constraint), so this requirement must be relaxed. However, the stability proof of SGPC and CSGPC depends on the property implied by the use of FIR's that the predicted output settles after n_y steps. Thus, an obvious strategy to follow is to: (i) retain this last property, but (ii) allow the value to which the *predicted* output settles at to be equal to some slack variable, s_∞, which will be a degree of freedom (DOF). The value to which the predicted output settles is determined by c_∞, the

far future command inputs; making c_∞ a DOF is the subject of section 4.1. Obviously, something must be done to ensure that the output goes to its target value; this is where the modifications differ. The first modification, described in Section 4.2, ensures that the output goes to its target value by augmenting the MWLS problem appropriately to include a penalty on the deviation of s_∞ from this target value. The modified algorithm is activated only when CSGPC is STIF, and guarantees recovery of short term feasibility (STF); this together with the properties of CSGPC guarantee stability and asymptotic tracking. Unfortunately, during the feasibility recovery stage, the modified algorithm does not place any penalty on the norm of the vector of predicted errors, and this may degrade transient performance. The other modifications, which are introduced below, improve on the first in that they do retain performance in the cost.

Recent work by Zheng and Morari [53] and Allwright [1] minimize the infinity-norm of the predicted errors rather than the two-norm as CSGPC does. Stability is guaranteed without recourse to terminal constraints, and therefore without the need for a STF assumption, however, the results are restricted to open-loop stable systems. Within the context of SGPC this work can be extended to unstable systems. The performance of such systems, though, is often not as good as CSGPC, as all effort is spent minimizing just the largest error, and this is often the first one. The system is driven very hard and the responses can be oscillatory. We, therefore, propose a procedure for dealing with STIF in which the objective is usually the standard CSGPC two-norm minimization, but when STIF is encountered, the value of the steady-state predicted output is allowed to become a DOF and the objective is shifted to the minimization of the infinity-norm of the predicted errors until STF is regained. Thus, by mixing objectives, the superior performance of CSGPC is retained when possible, but

stability and asymptotic tracking are guaranteed with only the assumption of long term feasibility (LTF). Section 4.3 develops an extension of the MWLS algorithm which performs the constrained optimization of an infinity-norm of the predicted errors and subsequently embeds this extension into the CSGPC framework. Finally, an algorithm which mixes these two objectives, Mixed-Objective CSGPC, is presented and illustrated by numerical example in section 4.4.

In Section 4.5 we propose a modification to CSGPC which shares in common with that of Section 4.2 the property that it is guaranteed to recover STF, but also includes in the objective a component which penalizes deviation of the predicted output values from desired set-point values; we will call this Modified Constrained Stable Generalized Predictive Control (MCSGPC). Unlike Mixed-Objective CSGPC, the performance cost is always a two-norm of the predicted errors, but when CSGPC is STIF, return to STF (and thus stability) is guaranteed by the added constraint that s_∞ move closer to the desired value at each successive time instant.

4.1 Adding an additional degree of freedom to CSGPC

STIF arises during set-point changes because of the implicit requirement that y should reach the new set-point in n_y steps and with only n_c "free" command inputs. All far future command inputs are chosen to be $c_\infty = r/b(1)$, where the reference signal, $r(t)$, is assumed, without loss of generality, to be a step of size r. Thus, the only way to overcome STIF is to relax this end-point constraint for the *predicted* output. A simple way to achieve this, while preserving the FLS attributes of CSGPC (which are used in the proof of stability), is to include c_∞ as a degree of freedom, such that the *predicted*

output settles at some slack variable, $s_\infty = b(1)c_\infty$ rather than r. To achieve this, eqn. (2.13) is modified to get:

$$\underset{\rightarrow}{y} = \Gamma_b^\blacklozenge \underset{\rightarrow}{c}^\blacklozenge + \underset{\rightarrow}{y_f}; \qquad \underset{\rightarrow}{\Delta u} = \Gamma_\alpha^\blacklozenge \underset{\rightarrow}{c}^\blacklozenge + \underset{\rightarrow}{\Delta u_f}; \qquad \underset{\rightarrow}{u} = \Gamma_a^\blacklozenge \underset{\rightarrow}{c}^\blacklozenge + \underset{\rightarrow}{u_f};$$
$$\underset{\rightarrow}{c}^\blacklozenge = \begin{bmatrix} \underset{\rightarrow}{c} \\ c_\infty \end{bmatrix}; \quad \Gamma_b^\blacklozenge = [\Gamma_b, \theta_b]; \quad \Gamma_\alpha^\blacklozenge = [\Gamma_\alpha, \theta_\alpha]; \quad \Gamma_a^\blacklozenge = [\Gamma_a, \theta_a] \tag{4.1}$$

Input constraints inequality (2.29) can then be rewritten as:

$$\| A^\blacklozenge \underset{\rightarrow}{c} - v^\blacklozenge(t) \|_\infty = \| e^\blacklozenge \|_\infty \le 1; \qquad A^\blacklozenge = \begin{bmatrix} \frac{1}{R}\Gamma_\alpha^\blacklozenge \\ \frac{1}{U}\Gamma_a^\blacklozenge \end{bmatrix} \quad v^\blacklozenge(t) = \begin{bmatrix} -\frac{1}{R}\underset{\rightarrow}{\Delta u_f} \\ \frac{1}{U}\left(U_o \underset{\rightarrow}{1} - \underset{\rightarrow}{u_f}\right) \end{bmatrix} \tag{4.2}$$

On the basis of (4.1) and (4.2), the CSGPC problem can be restated with respect to the augmented vector of future c's, $\underset{\rightarrow}{c}^\blacklozenge$, and these modifications can be deployed either all the time, or at all time instants when the original CSGPC algorithm runs into STIF.

The problem is that while c_∞ is a DOF, the predicted steady-state error is non-zero and thus the finite horizon cost is no longer equal to an infinite horizon cost, and therefore can no longer be shown to be a stable Lyapunov function. Alternative methods of guaranteeing stability must be used; three different methodologies are given below.

4.2 A stable constrained predictive control algorithm

While c_∞ is a DOF, the *predicted* output will settle at some value, $s_\infty = b(1)c_\infty$. Clearly, one needs to penalize the deviation of s_∞ from its desired value of r. The following algorithm applies CSGPC as described in Algorithm 2.1, but if at any time instant, $\| A\underset{\rightarrow}{c} - v(t) \|_\infty$ can not be made less than one, MWLS is reapplied with the following modifications:

$$\epsilon^{(i+1)} = S^{\blacklozenge}\left[\underset{\rightarrow}{c}^{\blacklozenge(i+1)} - \begin{bmatrix} \underset{\rightarrow}{c_o} \\ \frac{r}{b(1)} \end{bmatrix}\right]; \quad e^{\blacklozenge(i+1)} = A^{\blacklozenge}\underset{\rightarrow}{c}^{\blacklozenge(i+1)} - v^{\blacklozenge}(t); \quad S^{\blacklozenge} = \left[\mathbf{0}_{n_c}^{T} \;\; b(1)\right] \tag{4.3}$$

During the period of STIF, the monotonicity of return to STF is guaranteed by replacing the part of the cost that relates to performance, $S[\underset{\rightarrow}{c} - \underset{\rightarrow}{c_o}]$, with a penalty on the deviation of s_∞ from its desired value. Thus, while s_∞ is allowed to vary so as to make feasible solutions possible, MWLS will converge to the value closest to r which does not cause any constraint violations. The subsequent values of s_∞ will be shown to monotonically converge toward r until the problem is once again STF.

Algorithm 4.1 At each time instant t

Step 1: Apply CSGPC (Algorithm 2.1). If $\|A\underset{\rightarrow}{c} - v(t)\|_\infty \leq 1$, increment t by one and return to Step 1; otherwise proceed to Step 2.

Step 2: Use MWLS (Section 2.3) with the modifications of eqn. (4.3), let MWLS converge to the optimal $\underset{\rightarrow}{c}^{\blacklozenge}$, and implement the implied first future value of u. Increment t, go to Step 1.

Theorem 4.1 Under the assumption that the set-point changes do not cause the control problem to become infeasible (LTIF), Algorithm 4.1 has guaranteed stability and will cause the output y to reach asymptotically its target value.

Proof: If the problem is STF for all t, Step 2 will never be entered, so that Algorithm 4.1 will operate exactly as CSGPC, Algorithm 2.1, and so, by Theorem 2.2, we have stability and asymptotic tracking.

Now let us assume that at some t, CSGPC is STIF and Algorithm 4.1 enters Step

2. Clearly, the implied optimization problem is always feasible because c_∞ is now a degree of freedom and can be chosen so as to require as little movement in the u's, and Δu's as is necessary; hence, MWLS will converge to the solution which minimizes the distance of s_∞ from r, and satisfies inequality (4.2). The implied cost of Step 2 has exactly the same form as that of eqn. (2.30), and so, like the original cost, it will be monotonically decreasing; the arguments which prove this assertion are identical to those used for the original CSGPC algorithm (Theorem 2.2). Hence, Step 2 will cause s_∞ to assume its target value of r and will do so in the minimum number of time instants. Now, for $c_\infty = r/b(1)$, we have:

$$\| A^\blacklozenge \underset{\rightarrow}{c}^\blacklozenge - v^\blacklozenge(t) \|_\infty = \| A\underset{\rightarrow}{c} - v(t) \|_\infty \tag{4.4}$$

and so one time instant before c_∞ is made equal to $r/b(1)$, the CSGPC problem will become STF. This is so because, if Step 2 were applied one more time, it would give a vector $\underset{\rightarrow}{c}^\blacklozenge$ whose last element would be $r/b(1)$ and for which the quantity of eqn. (4.4) would be less than or equal to 1; hence, at that time instant it would be known that for $c_\infty = r/b(1)$ there exists a vector $\underset{\rightarrow}{c}$ which satisfies inequality (2.29). In conclusion, Step 2 will always recover STF, and hence, the overall algorithm will be stable and will cause y to asymptotically track its target. □

Example 4.1 In Example 3.1, we saw that at $t=2$ CSGPC, Algorithm 2.1 encountered STIF. Repeated application of the algorithm, as expected, led to instability. This problem is overcome entirely by the application of Algorithm 4.1 which, at $t=2$, invokes Step 2 and, therefore, results in a Δu_2 which is less than the maximum allowable size of 0.04 (see Figure 4.1c). This reduction of Δu_2 in turn results in a $b_{\Delta u}$ of smaller absolute

value (see Figure 4.1d) which lies within the interval of -0.0764 and 0.0764. From Figure 4.1c, it can be seen that all the u's used by the algorithm stay far from the absolute constraints and hence m_U and m_L will be large and $b_{\min}/b_{\max}$ will stay at -0.0764/0.0764 throughout the application of the algorithm. Then, from Figure 4.1b it is seen that the algorithm is feasible throughout, and recovers STF for CSGPC at t=25 (see Fig. 4.1d). The output response (Figure 4.1a) can be seen to be good.

4.3 ℓ_∞-CSGPC

If s_∞ is allowed to vary from r so $\|A\underset{\rightarrow}{c}-v(t)\|_\infty$ can be made less than one, then, as the predicted steady-state error is not zero, the finite horizon cost is no longer equal to the infinite horizon cost and thus the monotonicity proof for stability breaks down. If, however, the objective is shifted from minimizing the two-norm of the predicted errors to minimizing the infinity-norm, then monotonicity can be retained. In section 4.3.1, we define a CSGPC algorithm which has c_∞ as a degree of freedom and minimizes the infinity-norm of the predicted errors; we will call this ℓ_∞CSGPC. Then, in section 4.3.2, we modify MWLS to minimize an infinity-norm cost in the presence of constraints; the resulting algorithm shall be referred to as ℓ_∞-MWLS. Finally, in section 4.3.3, we give the stability properties of ℓ_∞-CSGPC.

4.3.1 The overall strategy of ℓ_∞-CSGPC

In ℓ_∞-CSGPC we make two changes to CSGPC. First, to overcome the problem of short term infeasibility, c_∞ is allowed to vary; this was done in section 4.1. Then, to ensure

that monotonicity of cost is retained, the objective is changed to the minimization of the infinity-norm of the predicted errors. The vector of predicted output errors, ε, is represented as:

$$\varepsilon_{n_1,n_y} = E^T(\underset{\rightarrow}{y} - \underset{\rightarrow}{r}) = H\underset{\rightarrow}{c}^{\blacklozenge} - f; \qquad E=[e_{n_1}, \ldots, e_{n_y}]; \qquad H = E^T\Gamma_b^{\blacklozenge}; \qquad f = E^T(\underset{\rightarrow}{r} - \underset{\rightarrow}{y}_f) \qquad \textbf{(4.5)}$$

where e_i is the i^{th} standard basis vector in $\Re^{n_y}$. Because $\underset{\rightarrow}{y}$ settles after n_c+n_b steps, we shall assume that, for ℓ_∞-CSGPC, $n_y=n_c+n_b+1$. The choice of n_1, the initial output horizon, determines the first predicted error to be included in the cost, J. Normally, n_1 can be set to one, and all future errors up to n_y, the output horizon, will be included in ε. When $b(z)$ exhibits non-minimum phase characteristics, though, n_1 must be chosen greater than one so that some of the initial (non-minimum phase transient) errors are ignored; the quantitative treatment of this problem will be presented in section 4.3.3.

The following algorithm is used to implement ℓ_∞-CSGPC:

Algorithm 4.2 (ℓ_∞-CSGPC) At each time instant t

Step 1: Calculate the vector of future command inputs which minimizes the infinity-norm of the predicted errors without violating the constraints, namely, $\min_{\underset{\rightarrow}{c}}. \; J_t = \| \varepsilon_{n_1,n_y} \|_\infty$ s.t. $\| A^{\blacklozenge}\underset{\rightarrow}{c}^{\blacklozenge} - v^{\blacklozenge}(t) \|_\infty \le 1$.

Step 2: Calculate and implement the first control increment using eqn. (4.1)b.

Step 3: Increment t by one and return to Step 1.

For convenience, the $^{\blacklozenge}$ superscript will be suppressed in sections 4.3.2 and 4.3.3, as this is the form (with c_∞ a DOF) assumed throughout the development of ℓ_∞-CSGPC.

4.3.2 The ℓ_∞-MWLS algorithm and its properties

The constrained minimization of an ℓ_∞-norm for the purpose of predictive control has been implemented in the literature with linear programming theory. Here, as an alternative, and to demonstrate the versatility of MWLS, we recast MWLS to achieve an infinity-norm objective rather than its original two-norm objective. In this section we derive the properties of ℓ_∞-MWLS.

Let the ℓ_∞-MWLS cost be:

$$J^{(i)}_{\ell_\infty\text{-}MWLS} = \left\| \begin{bmatrix} [w^{(i)}]^{1/2} \varepsilon^{(i)} \\ [W^{(i)}]^{1/2} e^{(i)} \end{bmatrix} \right\|_2^2; \qquad \varepsilon^{(i)}_{n_1,n_y} = H \underset{\rightarrow}{c}^{(i)} - f; \qquad e^{(i)} = A \underset{\rightarrow}{c}^{(i)} - v(t) \tag{4.6}$$

where the weights are updated at each iteration as per:

$$w^{(i+1)}{}_{jj} = \frac{w^{(i)}{}_{jj} \left| \varepsilon^{(i)}{}_j \right|}{\left\| w^{(i)} \varepsilon^{(i)} \right\|_1 \left\| W^{(i)} e^{(i)} \right\|_1} \sum_{k=1}^{n} w^{(i)}{}_{kk}; \qquad W^{(i+1)}{}_{jj} = \frac{W^{(i)}{}_{jj} \left| e^{(i)}{}_j \right|}{\left\| W^{(i)} e^{(i)} \right\|_1} \tag{4.7}$$

To gain an intuitive feel for the algorithm, it is useful to sum over j the weight updates of eqns. (4.7):

$$\sum_{j=1}^{n} w^{(i+1)}{}_{jj} = \frac{1}{\left\| W^{(i)} e^{(i)} \right\|_1} \sum_{k=1}^{n} w^{(i)}{}_{kk}; \qquad \sum_{j=1}^{m} W^{(i+1)}{}_{jj} = 1 \tag{4.8}$$

where the summation of the numerator of each equation in (4.7) is equal to, and therefore cancels, the one-norm in the respective denominator. The partition of the cost which contains ε is associated with performance, while the partition containing e is associated with constraint violations. At each iteration, the total weight, W, placed on the constraint partition is normalized to one, and the total weight, w, placed on the performance partition is adjusted by a measure of the size of the constraint violations. When constraints are violated, $\|e\|_\infty$ is greater than one, thus, less weight is placed on performance which forces the algorithm to emphasise the constraint violations; but when

all constraints are met with strict inequality, the weights on performance are increased, shifting the emphasis away from constraints. Within each partition, though, the largest elements are always emphasized. The following definitions will be used in the derivation of the properties of ℓ_∞-MWLS:

$$\begin{aligned} \underset{\rightarrow}{c}^{\#} &\equiv \lim_{i\to\infty} \underset{\rightarrow}{c}^{(i)} \quad \text{(if it exists)} \\ \underset{\rightarrow}{c}^{o} &\equiv \text{unconstrained } \ell_\infty \text{ optimum: } \min_{\underset{\sim}{c}} \|\varepsilon\|_\infty \\ \underset{\rightarrow}{c}^{*} &\equiv \text{constrained } \ell_\infty \text{ optimum: } \min_{\underset{\sim}{c}} \|\varepsilon\|_\infty \; s.t. \; \|e\|_\infty \le 1 \end{aligned} \tag{4.9}$$

Lemma 4.1 The ℓ_∞-MWLS feasibility region, F_{n_c}, is always non-empty for the disturbance free case ($F_{n_c} \neq \{0\}$).

Proof: ℓ_∞-MWLS has c_∞ as a degree of freedom, therefore, at start up, doing nothing is always a feasible solution, and thus, it can be argued recursively that the control trajectory calculated at the previous time instant will remain feasible in a disturbance free environment. □

Theorem 4.2 ℓ_∞-MWLS cannot converge to a point outside the feasibility region ($\underset{\rightarrow}{c}^{\#} \in F_{n_c}$).

Proof: Assume that ℓ_∞-MWLS converges to $\underset{\rightarrow}{c}^{\#} \notin F_{n_c}$, then $|e_k| > 1$ for $k \in I \subseteq I_o = \{1,2,\ldots,m\}$, where m is the total number of constraints. Let $\bar{I}$ be the complement of I with respect to I_o. Then, in the limit, $W_{jj}^{(\infty)} = 0$ for $j \in \bar{I}$, because $W_{kk}^{(i)}$ for $k \in I$ are increasing (relative to the others) with i, but, by eqn. (4.8)b, they all sum to one. Hence, because $\Sigma_I W_{kk}^{(i)} = 1$,

$$\left\| W^{(i)} e^{(i)} \right\|_1 = \sum_{k=1}^{m} W^{(i)}{}_{kk} \left| e^{(i)}{}_k \right| = \sum_{I} W^{(i)}{}_{kk} \left| e^{(i)}{}_k \right| > 1 \tag{4.10}$$

This, with eqn. (4.8)a, shows that the sum of the performance weights will decrease with i, and in the limit, $\Sigma w_k^{(\infty)}=0$. Therefore, the ℓ_∞-MWLS cost will become:

$$\lim_{i\to\infty} J^{(i)}_{\ell_\infty\text{-}MWLS} = \left\| [W^{(i)}]^{1/2} e^{(i)} \right\|_2^2 \tag{4.11}$$

whose solution is feasible since, by Lemma 4.1, $F_{n_c} \neq \{0\}$. This contradicts the assumption that $\underset{\rightarrow}{c}^{\#} \notin F_{n_c}$, and thus completes the proof. □

Corollary 4.1 If $\underset{\rightarrow}{c}^{o} \in F_{n_c}$, then $\underset{\rightarrow}{c}^{o} = \underset{\rightarrow}{c}^{*}$ and ℓ_∞-MWLS can only converge to $\underset{\rightarrow}{c}^{*}$ ($\underset{\rightarrow}{c}^{\#} = \underset{\rightarrow}{c}^{*}$).

Proof: $\underset{\rightarrow}{c}^{o}$ minimizes $\| \varepsilon \|_\infty$, and since $\underset{\rightarrow}{c}^{o} \in F_{n_c}$, the corresponding e is such that $\| e \|_\infty \leq 1$; therefore, $\underset{\rightarrow}{c}^{o} = \underset{\rightarrow}{c}^{*}$. Assume that ℓ_∞-MWLS converges to $\underset{\rightarrow}{c}^{\#} \neq \underset{\rightarrow}{c}^{*}$, but by Theorem 4.2, $\underset{\rightarrow}{c}^{\#} \in F_{n_c}$ and so $|e_k| = 1$ for $k \in I$ and $|e_j| < 1$ for $j \in \bar{I}$, where, as before, I is a subset of I_o and $\bar{I}$ is its compliment. Then, if $I \neq \{0\}$, $W_{jj}^{(\infty)} = 0$ for $j \in \bar{I}$, and

$$\left\| W^{(i)} e^{(i)} \right\|_1 = \sum_{k=1}^{m} W^{(i)}{}_{kk} \left| e^{(i)}{}_k \right| = \sum_{I} W^{(i)}{}_{kk} \left| e^{(i)}{}_k \right| = \sum_{I} W^{(i)}{}_{kk} = 1 \tag{4.12}$$

Therefore, by eqn. (4.8)a, the sum of the performance weights will be constant, $\Sigma w_k^{(i+1)} = \Sigma w_k^{(i)} = \gamma$. The individual weights will then be updated as per:

$$w^{(i+1)}{}_{jj} = \frac{w^{(i)}{}_{jj} \left| e^{(i)}{}_j \right|}{\left\| W^{(i)} \varepsilon^{(i)} \right\|_1} \gamma; \tag{4.13}$$

and so by Lawson [22], $\underset{\rightarrow}{c}^{\#} = \underset{\rightarrow}{c}^{*}$, which contradicts our earlier assumption that $\underset{\rightarrow}{c}^{\#} \neq \underset{\rightarrow}{c}^{*}$.

If $I=\{0\}$, $\| W^{(i)} e^{(i)} \|_1 < 1$, and the sum of the performance weights will increase with i. This has the effect of shifting all of the emphasis to performance as none of the constraints are active, and hence, we once again contradict our earlier assumption that

$\vec{c}^{\#} \neq \vec{c}^{*}$. Note that the growth of the performance weights does not constitute a numerical problem, because the algorithm can be stopped after $\vec{c}^{\#}$ converges and before the performance weights get too large. Alternatively, the sum of the performance and constraint weights can be normalized to one after the update of eqn. (4.7); then, if $I=\{0\}$, all the constraint weights would tend to zero. □

Theorem 4.3 ℓ_∞-MWLS can only converge to $\vec{c}^{*}$ ($\vec{c}^{\#}=\vec{c}^{*}$).

Proof: By corollary 4.1, this is true for $\vec{c}^{o} \in F_{n_c}$, so here, let $\vec{c}^{o} \notin F_{n_c}$. Assume that ℓ_∞-MWLS converges to $\vec{c}^{\#} \neq \vec{c}^{*}$. By Theorem 4.2, $\vec{c}^{\#} \in F_{n_c}$, and so the polyhedroid defined by $\|\varepsilon\|_\infty$=constant which passes through $\vec{c}^{\#}$ must either intersect F_{n_c}, or be tangential to it (for example, see Figure 4.2). Of these, the former cannot be true, because any point inside the intersection of the polyhedroid and F_{n_c} will result in a smaller cost for ℓ_∞-MWLS. Hence, ℓ_∞-MWLS will not converge to $\vec{c}^{\#}$ unless the polyhedroid through $\vec{c}^{\#}$ is tangent to F_{n_c}. This can only happen at $\vec{c}^{\#}=\vec{c}^{*}$. If the tangency is a point, then the solution is unique; however, if the tangency is a common edge (line, plane, or hyper-plane), then the solution is non-unique because any point along the common edge tangency will produce the same optimum cost. This is discussed in Remark 4.2. □

Remark 4.1 In some cases, the solution to $\min_c J_t = \|\varepsilon\|_\infty$ s.t. $\|e\|_\infty \leq 1$ is not unique. When this occurs in ℓ_∞-MWLS, there will be less than n_c active (non-zero) weights which are associated with linearly independent rows of H or A. This will cause the matrix $[H^T w^{1/2} \;\; A^T W^{1/2}]^T$ to be rank deficient, and means that not all degrees of freedom are used to minimize the ℓ_∞-MWLS cost. This non-uniqueness can be visualized by considering

the tangency mentioned in the proof of Theorem 4.3, which, in this case, will be a common edge (for example, see Figure 4.3). In practice, it is sufficient to stop the algorithm prior to the rank deficiency occurring, but an alternative solution would be to allow ℓ_∞-MWLS to run only until the active set is identified and constrain $\underset{\rightarrow}{c}$ to lie in it. The remaining degrees of freedom could then be used to minimize the next largest error.

Remark 4.2 Unlike LP, ℓ_∞-MWLS does not need a feasible starting point. Additionally, for many problems, an exact solution is not necessary. In this case, ℓ_∞-MWLS can provide a close solution very quickly. A reasonable termination criterion for ℓ_∞-MWLS is whether or not the cost changes are smaller than a small threshold value, ϵ. By choosing ϵ to be relatively large, say 0.01, ℓ_∞-MWLS will terminate in just a few iterations, thus providing a very quick estimate of the optimum solution. In applications where computation time during the sampling interval is at a premium, using ℓ_∞-MWLS with a termination criterion which incorporates a large ϵ may provide a much better option than waiting for LP to find a feasible starting point (a separate LP), and then converge to the exact solution. This is illustrated below.

The following example concerns the feasibility region and convergence properties of ℓ_∞-MWLS.

Example 4.2 Let the system of Example 3.1 be subject to input absolute and rate constraints (2.28) for which U_o=0, U=1, and R=0.1. Apply ℓ_∞-CSGPC, using Algorithm 4.2. Figure 4.4 through Figure 4.7 show the feasibility region (solid lines) and contours of equal $\|\varepsilon\|_\infty$ (dashed lines) at four time instants with n_c=1 so that $\underset{\rightarrow}{c}^\bullet$

has two elements; $\vec{c}^{\,o}$ is indicated by a 'o', and $\vec{c}^{\,\#}$ by a '$_*$'. In all instances, it is obvious that ℓ_∞-MWLS has converged to the optimum, per Theorem 4.2 ($\vec{c}^{\,\#}=\vec{c}^{\,*}$). At t=6, the progression of ε (labelled ep_i), w, e, and W through 50 iterations is shown in Figure 4.8, and their final values after 100 iterations are:

ε = [-0.8653 -0.7385 -0.6495 -0.6013]

w = [8.5776 0 0 0]

e = [0.6737 -1 -1 0.2132 0.1215 0.0674 -0.0326 -0.1326 -0.1113 -0.0992]

W = [0 0.6875 0.3125 0 0 0 0 0 0 0]

Note that the non-zero weights correspond to the largest ε_i's and to active constraints. In this case, one performance weight and (as is evident in Figure 4.4) two constraints weights are active.

At t=9 (Figure 4.5), the final errors and weights are:

ε = [-0.1620 0.0312 0.1234 0.1620]

w = [98.7436 0 0 17.8883]

e = [-0.8243 -1 -0.5758 0.2234 0.0974 -0.1635 -0.2635 -0.3211 -0.2987 -0.2890]

W = [0 1 0 0 0 0 0 0 0 0]

This time, two performance weights and (from Figure 4.5) one constraint weight are active.

At t=11 (Figure 4.6), only the first performance weight and the first constraint weight are active as indicated below:

ε = [0.0412 0.0269 -0.0224 -0.0398]

w = [121.31 0 0 0.0006]

e = [-1 0.9649 0.4421 -0.1203 -0.0437 -0.3631 -0.2666 -0.2224 -0.2344 -0.2388]

W = [1 0 0 0 0 0 0 0 0 0]

The corresponding rows of H and A are [1 0], and, therefore, not linearly independent; this is a case of a non-unique solution.

Finally, at $t=14$ (Figure 4.7), the unconstrained optimum is feasible, and as stated in Corollary 4.1, $\vec{c}^{\,o}=\vec{c}^{\,\#}=\vec{c}^{\,*}$. The final errors and weights are:

ε = [-0.0040 0.0040 0.0040 -0.0040]

w = 1.0e+024 * [0.6080 0.0715 1.1839 0.8822]

e = [-0.5883 -0.2036 0.2380 0.0019 -0.0204 -0.2493 -0.2696 -0.2458 -0.2457 -0.2477]

W = [1 0 0 0 0 0 0 0 0 0]

Note that no constraints are active, and all degrees of freedom are used to minimize the performance cost. The performance weights have grown large, but this did not adversely affect the algorithm and could have been averted by appropriate normalization.

To illustrate the time savings possible with ℓ_∞-MWLS, the minimization in Step 1 of Algorithm 4.2 was run with ℓ_∞-MWLS (using the termination criterion mentioned in Remark 4.2, ie: setting $\epsilon=0.01$) and with LP. The former never needed more than 11 iterations, taking a maximum of 0.11 seconds; LP required a maximum of 0.33 seconds. The use of a "relatively large" value for the threshold ϵ results in a slight degradation in performance (the ℓ_∞-MWLS cost is about 1% larger than the LP cost),

but this is a small price to pay when considering the very significant reduction in computation time (threefold).

4.3.3 The stability of ℓ_∞-CSGPC

Once the optimum vector of future reference values, $\vec{c}^*$, has been calculated by ℓ_∞-MWLS, the procedure for computing and implementing the first optimum control increment is exactly the same as that used in CSGPC [33]. The arguments for the stability of ℓ_∞-CSGPC are also similar, as presented here.

Lemma 4.2 Let a linear system with transfer function $G(z)=z^{-1}b(z)/a(z)$ be subject to input constraints which, at time t and for reference horizon n_c and input horizon $n_u > n_a+n_c+1$, are given as $\| A\vec{c}-v(t) \|_\infty \le 1$. Furthermore, let the ℓ_∞-CSGPC objective be $\min_c J_t = \| \varepsilon_{n_1,n_y} \|_\infty$ s.t. $\| A\vec{c}-v(t) \|_\infty \le 1$, then for $1 \le n_1 \le n_y$, J_t is non-increasing for all t and ℓ_∞-CSGPC is BIBO stable.

Proof: First, we note that $\| \varepsilon_{n_1,n_y} \|_\infty = \| \varepsilon_{n_1,\infty} \|_\infty$, because, by the property of FIR's and the choice of $n_y > n_b+n_c+1$, the predicted output will reach steady-state at the n_y^{th} step. At each new time instant, $t+1$, the control trajectory implied by the previous optimum is feasible and can be used to give a cost, J_{t+1}, which will be less than or equal to the optimum cost J_t^* at t. Then, the optimum at $t+1$, J_{t+1}^*, will be less than or equal to J_{t+1}, and hence, less than or equal to J_t^*. □

Lemma 4.3 For $n_1=n_y$, ℓ_∞-CSGPC is stable and gives asymptotic tracking.

Proof: The objective is now equivalent to that of Step 2 in Algorithm 4.1, which is used

when CSGPC encounters STIF. □

Lemma 4.4 If ℓ_∞-CSGPC is at rest, u_{ss} is on the interior of the absolute constraint limits, and $y_{ss} \neq r$, then there exists an n_1 such that $\| \varepsilon_{n_1,n_y} \|_\infty$ can be reduced without violating the constraints.

Proof: The elements of this steady-state error vector will be $\varepsilon_1 = \varepsilon_2 = \cdots = \varepsilon_{n_y} \equiv \varepsilon_{ss} = r - y_{ss} \neq 0$. Furthermore, $\| A\vec{c}_{ss} - v(t) \|_\infty$ must be strictly less than one because the rates are inactive and the problem is long term feasible (LTF). Therefore, there exists a positive number ϵ such that $\| A(\vec{c}_{ss} + \delta c) - v(t) \|_\infty \leq 1$ for all δc which satisfy $\| \delta c \|_\infty \leq \epsilon$. Now then, we need to show that there exists a δc which satisfies:

$$\| H(\vec{c}_{ss} + \delta c) - f \|_\infty = \| H\delta c + (H\vec{c}_{ss} - f) \|_\infty = \| H\delta c + \varepsilon_{ss}\mathbf{1} \|_\infty < \| H\vec{c}_{ss} - f \|_\infty = | \varepsilon_{ss} | \qquad \textbf{(4.14)}$$

and $\| \delta c \|_\infty \leq \epsilon$. This can be done easily by choosing $n_1 > n_b$ such that $H\mathbf{1} = b(1)\mathbf{1}$ and setting $\delta c = -\text{sign}\{b(1)\}\epsilon\mathbf{1}$ so that:

$$\| H\delta c + \varepsilon_{ss}\mathbf{1} \|_\infty = \varepsilon_{ss} - |b(1)|\,\epsilon < | \varepsilon_{ss} | \quad \text{and} \quad \| \delta c \|_\infty = \epsilon \qquad \textbf{(4.15)}$$

Eqn. (4.15)a will be true, because one would always choose $\epsilon \leq \varepsilon_{ss}/|b(1)|$; $\delta c = -\varepsilon_{ss}\mathbf{1}/b(1)$ will make $y_{ss} = r$. □

Theorem 4.4 There exists an n_1 such that:

$$\min_x \| Hx - \mathbf{1} \|_\infty < 1 \qquad \textbf{(4.16)}$$

Furthermore, if a system is at rest, u_{ss} is on the interior of the absolute constraint limits, and $y_{ss} \neq r$, then ℓ_∞-CSGPC will remain at rest at the wrong steady-state value if, and only if, n_1 is chosen such that condition (4.16) is not satisfied.

Proof: That such an n_1 exists is proved by choosing $n_1 > n_b$ which makes the LHS of

condition (4.16) equal to zero. Now, $\|Hx\text{-}\mathbf{1}\|_\infty < 1$ is equivalent to condition (4.14) with ε_{ss} normalized to 1, so if n_1 is chosen such that condition (4.16) is not satisfied, no control trajectory will exist which produces a lower cost than that of remaining at rest ($\delta c = \mathbf{0}$), but if condition (4.16) is satisfied, then such a trajectory does exist and can be made small enough to keep $\|A(\underset{\rightarrow}{c}+\delta c)-v(t)\|_\infty \leq 1$. The arguments for this are similar to those given in the proof of Lemma 4.4. ℓ_∞-CSGPC will choose the optimum cost, so in the first instance, y will remain at rest at the wrong steady-state value, and in the second, it will move towards the desired set-point. □

Remark 4.3 Note, in Lemma 4.3 ($n_1 = n_y$), that only the last error, ε_{n_y}, is minimized. But, to maintain as much of the transient behaviour in the objective as is possible, we want n_1 as small as possible. Condition (4.16) can be used to determine the smallest valid n_1. By starting with $n_1 = 1$, condition (4.16) can be tested (off line, with a linear program or Lawson's algorithm) and n_1 can then be incremented until the condition is satisfied. Hereafter, we define n_1^* to be the smallest n_1 which satisfies condition (4.16).

Remark 4.4 Theorem 4.4 states that if n_1 is chosen less than n_1^*, then ℓ_∞-CSGPC will not track a reference; this would normally be associated with non-minimum phase attributes. Because of the SGPC stabilizing loop and by eqn. (2.13)a, this behaviour is determined solely by the system's numerator polynomial, $b(z)$. If $b(z)$ is such that the output must initially go further from the desired steady-state value before it can go closer (ie. non-minimum phase behaviour), then the first error(s) will always be greater than they would be if the system just stayed at rest. Thus, since ℓ_∞-CSGPC minimizes the maximum error, this initial error(s) must be excluded, or the "optimum" cost will be that

which requires the system to stay at steady-state.

As stated in Remark 4.2, the solution to $\min_c J_t = \|\varepsilon\|_\infty$ s.t. $\|e\|_\infty \le 1\}$ can be non-unique; therefore, the possibility of an undamped oscillating cost exists. To account for this, Theorem 4.5 is written with an assumption of uniqueness, but then the assumption is removed in Corollary 4.2 by modifying the cost objective to emphasize later errors more than earlier ones.

Theorem 4.5 Let the control trajectory which optimizes the cost of Lemma 4.2 be unique, then, for $n_1 \ge n_1^*$, ℓ_∞-CSGPC is stable and gives asymptotic tracking.

Proof: If $y_{ss}=r$, then we are tracking the reference, so let $y_{ss} \ne r$. Now, by Lemma 4.2, $J_{t+i}^* \le J_t^*$; if J_{t+i}^* has not stabilized at a constant non-zero value then the cost is decreasing as desired, so assume it has stabilized (ie. $J_{t+i}^* = J_t^*$ for all i). Uniqueness then implies that the control and output trajectories are frozen, and we have steady-state after $i = n_c + n_b$ steps. But, by Lemma 4.4, this cost can always be reduced, which contradicts the assumption of a stabilized non-zero cost. Therefore, the cost will be reduced to zero and ℓ_∞-CSGPC will asymptotically track the reference. □

Corollary 4.2 Let the ℓ_∞-CSGPC objective be $\min_c J_t = \|S\varepsilon_{n_1,n_y}\|_\infty$ s.t. $\|e\|_\infty \le 1$, where the elements of S are chosen such that $S_{jj} > S_{ii}$ for $j > i$ and $S_{ij}=0$ for $j \ne i$. Then, for $n_1 \ge n_1^*$, ℓ_∞-CSGPC is stable and gives asymptotic tracking.

Proof: If we use the same control trajectory at $t+1$ as we used at t, then each ε_i is multiplied by an "earlier" (ie. smaller) S_{ii}, so $J_{t+1} < J_t^*$. This will always be true unless the biggest ε_i at t is the last one (in such a case, the biggest at the next instant will also

be equal to $S_{n_y n_y}\varepsilon_{n_y}$). But then the extra degree of freedom which is available at the next time instant will be used to decrease J_{t+1}. This can always be done without causing any of the "earlier" $S_{i-1}\varepsilon_i$'s to become dominant as they are all strictly less than $S_{n_y n_y}\varepsilon_{n_y}$. Therefore, J_t is a monotonically decreasing function of time. □

Remark 4.5 S can also be used to increase the speed of convergence by increasing the emphasis on steady-state errors; in the limit one gets that part of Algorithm 4.1, which is used when CSGPC encounters STIF.

ℓ_∞-CSGPC gives stability by minimizing the maximum predicted error, as do the works of Zheng and Morari [53] and Allwright [1], but by first applying the SGPC stabilizing loop, ℓ_∞-CSGPC can also handle open-loop unstable systems. Unfortunately, min-max controllers often give undesirable performance because the maximum error is often the first (transient) one. This causes the system to be driven very hard and can lead to under-damped oscillatory response. CSGPC, on the other hand, minimizes the two-norm of the predicted errors, and therefore offers excellent performance, but cannot handle STIF. Thus, in the next section, we propose an algorithm which mixes these two objectives.

4.4 Mixed-Objective CSGPC

Mixed-Objective CSGPC offers a compromise between CSGPC and ℓ_∞-CSGPC. When STF, the preferred two-norm objective of CSGPC will be used, but when CSGPC is STIF, c_∞ will be made a DOF and the objective will be changed to the infinity-norm

objective of ℓ_∞-CSGPC until STF is regained for CSGPC.

Consider the following algorithm, which we shall term Mixed-Objective CSGPC.

Algorithm 4.3 (Mixed-Objective CSGPC) At each time instant t

Step 1: Test for short term feasibility, namely, that $\min_c \| A\underset{\rightarrow}{c} - v(t) \|_\infty \leq 1$; this can be done with either Lawson's algorithm or linear programming. If short term feasible, proceed to Step 2; otherwise, go to Step 3.

Step 2: Apply CSGPC as described in Algorithm 2.1, namely, $\min_c J_t = \| \epsilon \|_2^2$ s.t. $\| A\underset{\rightarrow}{c} - v(t) \|_\infty \leq 1$, $c_\infty = r/b(1)$; increment t and return to Step 1.

Step 3: Apply ℓ_∞-CSGPC as described in Algorithm 4.2, namely, $\min_c . \; J_t = \| S\varepsilon_{n_1,n_y} \|_\infty$ s.t. $\| A^\bullet \underset{\rightarrow}{c}^\bullet - v^\bullet(t) \|_\infty \leq 1$ with $n_1 \geq n_1^*$; increment t and return to Step 1.

Theorem 4.6 Mixed-Objective CSGPC has guaranteed stability and will cause the output y to reach asymptotically its target value.

Proof: If the problem is STF for all t, the modified algorithm will operate as CSGPC Algorithm 2.1, and so by Theorem 2.2, we have stability and asymptotic tracking. If, however, due to a set-point change, CSGPC is STIF, ℓ_∞-CSGPC will be applied. Now, by Lemma 4.1, it is known that this optimization problem will always be feasible, and furthermore, by Theorem 4.4, we have that the ℓ_∞-CSGPC cost is a monotonically decreasing function of time. Hence the output will be driven towards its target value, r. At some time instant before the output reaches r, CSGPC will regain STF and the algorithm will revert to CSGPC. Namely, by the results of section 4.3.3, we know that ℓ_∞-CSGPC will always recover STF for CSGPC, and hence, Mixed-Objective CSGPC will be stable and will cause the output to asymptotically track its target value. □

Now we give two numerical examples to illustrate the results of this and the previous section. In the first example, we show how ℓ_∞-CSGPC, using Algorithm 4.2, and Mixed-Objective CSGPC, using Algorithm 4.3, actually maintain stability. This example also serves to illustrate the sometimes oscillatory response of ℓ_∞-CSGPC and the great improvement achieved by Mixed-Objective CSGPC.

Example 4.3 Let the system of Example 3.1 be subject to constraints (2.28) for which $U_o=0$, $U=25$, and $R=0.04$. Setting $n_c=3$, the two algorithms are applied to control the system. Both ℓ_∞-CSGPC (Figure 4.9) and Mixed-Objective CSGPC (Figure 4.10) maintain stability and set-point tracking. Notice, though, that ℓ_∞-CSGPC gives an oscillatory response, taking 23 time steps to settle; this is because it always uses the infinity-norm objective. Mixed-Objective CSGPC, on the other hand, regained STF for CSGPC and reverted to the two-norm objective after only four time steps; the response is, therefore, very good.

As mentioned in section 4.3.3, infinity-norm controllers minimize the maximum predicted error, so for systems which exhibit non-minimum phase behaviour some of the initial errors must be ignored. We are concerned with the non-minimum phase behaviour of the closed-loop system, and as noted in Remark 4.4, this is determined solely by the open-loop system's numerator polynomial, $b(z)$. The last example illustrates this point.

Example 4.4 Let the system with transfer function $g(z)=z^{-1}b(z)/a(z)$, such that:

$$a(z) = 1-3.8z^{-1}+3.87z^{-2}-0.27z^{-3}-0.54z^{-4}$$
$$b(z) = 1-1.13z^{-1}-5.003z^{-2}+6.6378z^{-3} \quad \textbf{(4.17)}$$

be subject to input absolute and rate constraints (2.28) for which $U_o=0$, $U=0.1$, and $R=0.05$. This system is particularly difficult to control as it has a near pole-zero cancellation outside the unit circle; $a(z)$ has a root at 1.5, and $b(z)$ has roots at 1.55 and -3.0. For $n_c=3$ and $n_1=1$, the H matrix is:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.95 & 1 & 0 & 0 \\ -5.375 & 0.95 & 1 & 0 \\ 2.325 & -5.375 & 0.95 & 1 \\ 0 & 2.325 & -5.375 & 1.95 \\ 0 & 0 & 2.325 & -3.425 \\ 0 & 0 & 0 & -1.1 \end{bmatrix}$$

Now, for $n_1 \leq 2$, $\min_x \| \boldsymbol{Hx}-\mathbf{1} \|_\infty = 1$, and in fact the minimizing vector is $\boldsymbol{x}=\mathbf{0}$; clearly, condition (4.16) is violated, and ℓ_∞-CSGPC never moves from an initial rest condition. But for $n_1=3$, $\min_x \| \boldsymbol{Hx}-\mathbf{1} \|_\infty = 0.0546$, and the minimizing $\boldsymbol{x}$ is -[0.4717, 0.7037, 0.8124, 0.8594]^T; condition (4.16) is satisfied, and ℓ_∞-CSGPC tracks the reference as desired (Figure 4.11). For completeness, Figure 4.12 shows the response of Mixed-Objective CSGPC; in this case, the improvement is only minor.

These examples demonstrate the ability of ℓ_∞-CSGPC and Mixed-Objective CSGPC to handle open-loop unstable and non-minimum phase systems. Mixed-Objective CSGPC produces superior performance by retaining the two-norm objective whenever possible, but falling back to an infinity-norm objective when STIF is encountered.

4.5 Modified CSGPC

Here we propose an improved modification to CSGPC which retains the part of the cost that relates to performance as a two-norm; this provides for a more optimal choice of control moves during the return to short term feasibility. This modification always leads to stability and takes y to its set-point by requiring the deviation of s_∞ from r to decrease at each successive time instant while STF is being regained.

4.5.1 The MCSGPC algorithm

The strategy of setting $s_\infty = b(1)c_\infty$ as close as possible to r when CSGPC is STIF (Algorithm 4.1) makes good sense in that it drives the problem as close to CSGPC feasibility as is possible, thereby hastening the return of CSGPC to STF (we will call this value of s_∞, which is closest to r and which does not violate the constraints, s_∞^o). However when CSGPC is STIF, Algorithm 4.1 pays no attention to transient performance in that Step 2 ignores the cost J of eqn. (2.16). To overcome this problem, one should be looking for ways of retaining the minimization of J while at the same time guaranteeing the convergence of CSGPC to STF, namely guaranteeing that s_∞ converges to r. This can be achieved by adding the extra constraint that the current value of s_∞ should be closer to r than is s_∞^{old}, which is the value of s_∞ used at the previous step:

$$|s_\infty - r| \le \rho |s_\infty^{old} - r| \qquad \rho^o \le \rho < 1; \quad \rho^o = \frac{|s_\infty^o - r|}{|s_\infty^{old} - r|} \tag{4.18}$$

where the lower bound on ρ is ρ^o rather than zero to ensure that this new constraint is always feasible; it must be remembered that s_∞^o is the closest that we can get to r and this will be required when $\rho = \rho^o$ for which (4.18)a can only hold with equality. Thus

constraint (4.18) would provide the guarantee of convergence of CSGPC to STF that we are looking for. However ρ^o would vary from sampling instant to sampling instant, thereby possibly necessitating a change in the value of ρ at each sampling instant. A convenient (equivalent) alternative which avoids this problem is stated below; we will term this a slack variable end-point constraint.

Lemma 4.5 Given that at time t the vector of future values of c is such that no input constraint is violated, then at $t+1$ the constraint

$$|s_\infty - s_\infty^o| \leq \rho |s_\infty^{old} - s_\infty^o| \quad 0<\rho<1 \tag{4.19}$$

will be compatible with constraints (2.29).

Proof: If the control law implied by the vector of future values of c of the lemma is not changed at $t+1$, then the input constraints will again not be violated; thus $s_\infty{}^{old}$ represents a feasible choice for s_∞. On the other hand, by definition $s_\infty{}^o$ gives an alternative feasible choice for s_∞. Given the convex nature of the input constraints any choice of s_∞ between $s_\infty{}^{old}$ and $s_\infty{}^o$ will be consistent with the input constraints. This is exactly the range of choices allowed by constraint (4.19) as ρ tends to 1. □

The important point about condition (4.19) is that the larger ρ is chosen to be, the less stringent the requirement on s_∞ becomes and thus the more design freedom is released for other control purposes; this is in direct contrast to what is done in Algorithm 4.1 which corresponds to what would happen if ρ were chosen to be arbitrarily small. The design freedom which is made available by choosing ρ greater than zero can be deployed in the optimization of the tracking of the desired set-point r through the minimization of the cost J of eqn. (2.16).

To implement this modification using MWLS, input constraints inequality (4.2) is augmented to include this new constraint:

$$\| A^* \underset{\rightarrow}{c}^\blacklozenge - v^*(t) \|_\infty = \| e^* \|_\infty \leq 1; \quad A^* = \begin{bmatrix} \frac{1}{R}\Gamma_\alpha^\blacklozenge \\ \frac{1}{U}\Gamma_a^\blacklozenge \\ \hline \mathbf{0}_{n_c}^T \quad \frac{b(1)}{\rho(s_\infty^{old} - s_\infty^o)} \end{bmatrix} \quad v^*(t) = \begin{bmatrix} -\frac{1}{R}\Delta \underset{\rightarrow}{u}_f \\ \frac{1}{U}\left(U_o \underset{\rightarrow}{1} - \underset{\rightarrow}{u}_f\right) \\ \hline \frac{s_\infty^o}{\rho(s_\infty^{old} - s_\infty^o)} \end{bmatrix} \tag{4.20}$$

and MWLS is then applied with the following modifications:

$$\begin{aligned} \epsilon^{*(i+1)} &= S^* \left(\underset{\rightarrow}{c}^{\blacklozenge (i+1)} - \underset{\rightarrow}{c}_o^{\,*} \right); \qquad e^{*(i+1)} = A^* \underset{\rightarrow}{c}^{\blacklozenge (i+1)} - v^*(t) \\ S^{*2} &= \Gamma_b^{\blacklozenge T} \Gamma_b^\blacklozenge + \lambda \Gamma_\alpha^{\blacklozenge T} \Gamma_\alpha^\blacklozenge; \qquad \underset{\rightarrow}{c}_o^{\,*} = S^{*-2} [\Gamma_b^{\blacklozenge T} (\underset{\rightarrow}{r} - \underset{\rightarrow}{y}_f) - \lambda \Gamma_\alpha^{\blacklozenge T} \Delta \underset{\rightarrow}{u}_f]; \end{aligned} \tag{4.21}$$

Algorithm 4.4 (MCSGPC) At each time instant t

Step 1: $\min_{\underset{\rightarrow}{c}} |r - s_\infty|$ subject to constraints (2.29) and let the minimizing value of s_∞ be $s_\infty{}^o$ (use MWLS as per Algorithm 4.1 or a linear program).

Step 2: If $s_\infty{}^o = r$, CSGPC is STF; apply CSGPC (Algorithm 2.1) and go to step 4.

Step 3: For $s_\infty{}^o \neq r$, CSGPC is STIF; use MWLS (Section 2.3) with the modifications of eqn. (4.21).

Step 4: Implement the first element of $\underset{\rightarrow}{u}$, increment t, and go to step 1.

Remark 4.6 If CSGPC remains STF then Step 3 of MCSGPC will never be entered into and thus MCSGPC will reduce to CSGPC.

4.5.2 Convergence and stability of MCSGPC

In this section we prove that MCSGPC is stable and has guaranteed asymptotic set-point tracking.

Lemma 4.6 Given STF at start up, MCSGPC will remain STF at all subsequent times irrespective of the size of future set-point changes.

Proof: At all times that CSGPC is STF, MCSGPC will behave like CSGPC and will therefore retain STF. At times when a set-point change causes CSGPC to be STIF, MCSGPC reverts to Step 3 which, by the assumption of initial feasibility and Lemma 4.5, will be feasible and will remain so. □

Henceforth, it will be assumed that MCSGPC is feasible at start up and therefore at all subsequent times.

Lemma 4.7 At every time instant, $s_\infty^{\,o}$ will satisfy the condition:

$$|s_\infty^{o} - r| \leq |s_\infty^{old} - r| \tag{4.22}$$

that is, $s_\infty^{\,o}$ will be at least as close to r as was $s_\infty^{\,old}$.

Proof: Due to MCSGPC feasibility, it is known that $s_\infty^{\,old}$ was achievable at the previous time instant and therefore is reachable now, hence $s_\infty^{\,o}$, which by definition represents the closest achievable value to r, must be at least as near to r as is $s_\infty^{\,old}$. □

Theorem 4.7 MCSGPC has guaranteed stability and asymptotic set-point tracking.

Proof: At any time step when $s_\infty^{\,o}=r$, MCSGPC will revert to CSGPC (as per Step 2) and will thus be stable and track r asymptotically. So it remains to consider the case

when CSGPC is STIF, namely $s_\infty{}^o \neq r$. From Lemma 4.7, we distinguish two cases:

(i) $|s_\infty^o - r| < |s_\infty^{old} - r|$: By constraint (4.19), the convexity of input constraints, and the definition of $s_\infty{}^o$, we can show

$$|s_\infty^o - r| \leq |s_\infty - r| < |s_\infty^{old} - r| \tag{4.23}$$

which states that s_∞ (and $s_\infty{}^o$) is converging to r, thereby returning the problem to the case where CSGPC is STF.

(ii) $|s_\infty^o - r| = |s_\infty^{old} - r|$: This implies that $s_\infty{}^o$ and $s_\infty{}^{old}$ are equidistant from r. But the interval between $s_\infty{}^o$ and $s_\infty{}^{old}$ cannot contain r because, by convexity, then r would be achievable thereby implying $s_\infty{}^o = r$; this corresponds to the case when CSGPC is STF which has been examined earlier in the proof and is excluded here. Hence $s_\infty{}^{old} = s_\infty{}^o \neq r$ and by (4.19) s_∞ itself must equal $s_\infty{}^o$ and therefore $s_\infty{}^{old}$. This situation however can not persist for more than $n_a + n_c$ steps, because by then the system will have reached steady-state which implies that we could no longer be on the boundaries of any of the input constraints and $s_\infty{}^o$ could be brought closer to r, which brings us back to case (i).

Thus, if at any time step CSGPC is STIF, it cannot remain in case (ii) but will always return to case (i) and converge to the case where CSGPC is STF, which, as stated, implies stability and asymptotic tracking. □

Remark 4.7 Implicit in the proofs of this section is the assumption that the set-point has been chosen sensibly, ie. such that $u_{ss} = a(1)r/b(1)$ satisfies absolute constraint (2.28)b. It is easy to show, though, that should the set-point be chosen so that $a(1)r/b(1)$ is outside the constraint interval, then MCSGPC will cause s_∞ (and thus the output) to settle at a value dictated by whichever absolute constraint boundary $a(1)r/b(1)$ exceeds.

Remark 4.8 By Theorem 4.7, we have that MCSGPC will converge to where CSGPC is STF for all values of ρ, $0<\rho<1$; the particular choice of ρ can be used to strike the desired balance between speed of return of CSGPC to STF and the emphasis on transient tracking performance. The smaller ρ, the faster convergence of CSGPC to STF will be, but the less attention will be paid to performance; in the limit (for ρ arbitrarily small) MCSGPC reduces to Algorithm 4.1. Also, a low value of ρ minimizes the difference in the objectives of Steps 2 and 3 of MCSGPC. However, if one wishes to maintain the emphasis on cost optimization throughout, then one might wish to invoke Step 3 at all times (whether CSGPC is STF or not) and assign to ρ a value close to 1.

In summary, the algorithm presented in this section has two significant contributions to make: (i) it is guaranteed to retain STF; and (ii) it optimizes a two-norm of *actual* predicted tracking errors. Of course, in common with normal practice (and in agreement with common sense), STF must be assumed at start up, but in contrast to earlier work, no further feasibility assumptions are required.

4.5.3 Illustrative examples

In this section we give two numerical examples. The first illustrates the efficacy of MCSGPC (Algorithm 4.4) in dealing with set-point changes which are large enough to destabilize CSGPC and draws comparisons between the performance of MCSGPC and that of Algorithm 4.1. MCSGPC outperforms Algorithm 4.1; this is expected since MCSGPC, unlike Algorithm 4.1, uses the slack variable only to condition the end-point constraint, but deploys the actual set-point in the cost. The approach of making s_∞ a DOF and using a slack variable end-point constraint can also be used in the context of

algorithms with infinite output-constraint horizons such as that given in [30] (which we will term RM) to significant advantage as illustrated in the second example. In particular the example shows that freeing s_∞ and applying a slack variable end-point constraint overcomes the infeasibility and resulting instability problems of RM, and indeed leads to a good output response.

Example 4.5 Just as did Algorithm 4.1 in Example 4.1, MCSGPC, Algorithm 4.4, maintains stability and set-point tracking (Figure 4.13), but when compared by total cost,

$$J_{run} = \sum_{i=0}^{runtime} (r_{t+i} - y_{t+i})^2 + \lambda(\Delta u_{t+i-1})^2 \tag{4.24}$$

Algorithm 4.1 has a cost of 1.19, while the cost for MCSGPC is only 0.68. This is because Algorithm 4.1 neglects performance while CSGPC is STIF so as to drive s_∞ to r as quickly as possible; the performance is thus sub-optimal. MCSGPC, on the other hand, is guaranteed to retain STF and to converge to where CSGPC is STF at a rate dictated by the choice of ρ; all remaining freedom is then used to optimize performance. Hence, by moving s_∞ more judiciously (slowly) than Algorithm 4.1, it has improved performance. It is interesting to note that MWLS, with the modifications of eqn. (4.21), converged with much fewer iterations than it did with the modifications of eqn. (4.3); this appears to be a typical trait.

Example 4.6 Let the numerator and denominator polynomials be given by:

$$a(z) = 1 - 1.3z^{-1} + 0z^{-2} + 0.144z^{-3}; \qquad b(z) = 2 + 0.45z^{-1} + z^{-2} \tag{4.25}$$

which has an unstable pole at 1.2. Furthermore, let the input constraints be U_o=0, U=0.05 and R=0.01, and let the control parameters be n_c=3, λ=1, n_r=6, and ρ=0.99.

All algorithms are started with zero initial conditions and zero set-point; at $t=16$, the set-point is changed to one. The simulation results for this example are given in Figure 4.14 which depicts the output, slack variable (s_∞), input, and incremental input.

MCSGPC (dashed-dotted lines) once again can be seen to perform well, whereas RM (dashed lines) runs into instability problems. As explained earlier, the slack variable end-point constraint gives guaranteed stability and can be applied in the context of algorithms with infinite output horizons such as RM. This indeed can be seen to be the case from the solid line plots of Figure 4.14 which represent the results of RM with a slack variable end-point constraint. Our approach not only overcomes instability but also produces good transient and steady-state responses.

4.6 Chapter summary

A major problem in control engineering is guaranteeing stability in the presence of system input constraints. Predictive control provides a natural framework for handling constraints and recently has been adapted to give a guarantee of stability. However, all stability results are dependent on a feasibility assumption. In this chapter, we presented three modifications which guarantee the retention of feasibility (and stability) for any set-point change. The first modification has the advantage of simplicity, but ignores transient errors. The final two modifications, unlike other approaches which use set-point conditioning ([2], [14]), retain the actual set-point in the cost and therefore are able to retain optimality of tracking; in both Mixed-Objective CSGPC and MCSGPC, the slack-variable conditions the end-point constraint only.

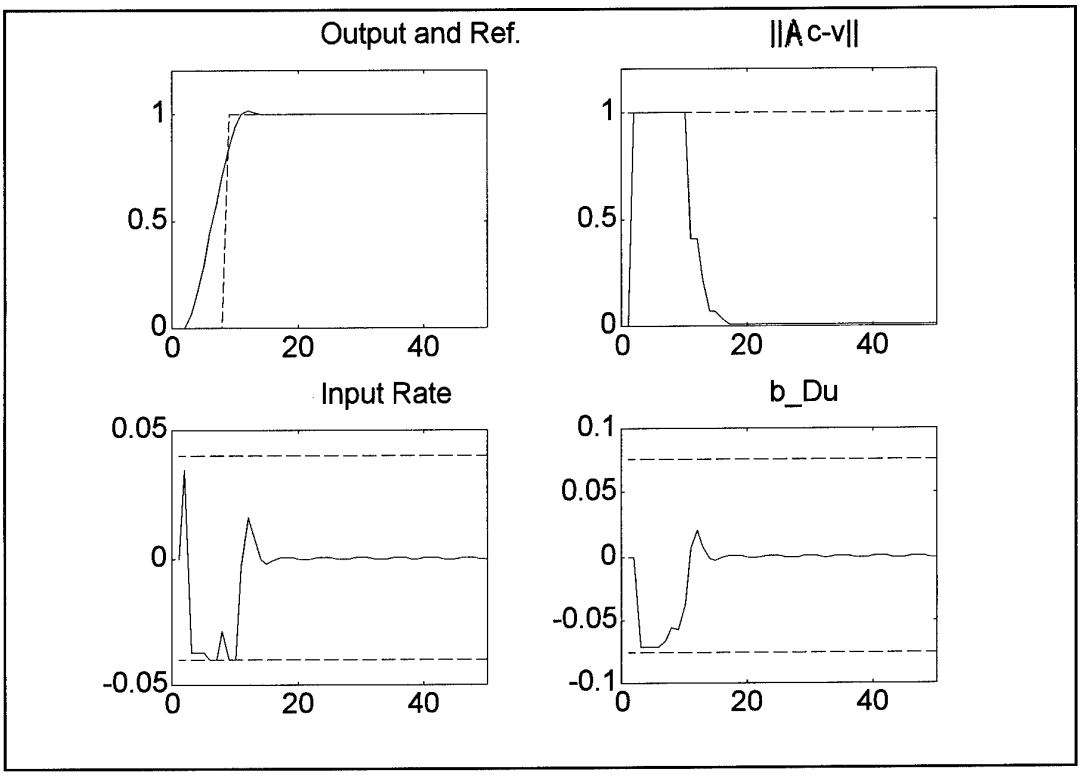


Figure 4.1 Example 4.1 - Algorithm 4.1 response

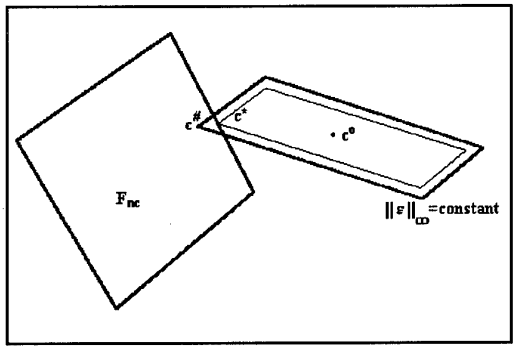


Figure 4.2 A contradictory Solution

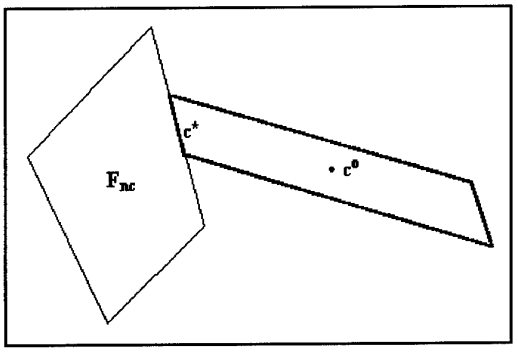


Figure 4.3 A Non-unique Solution

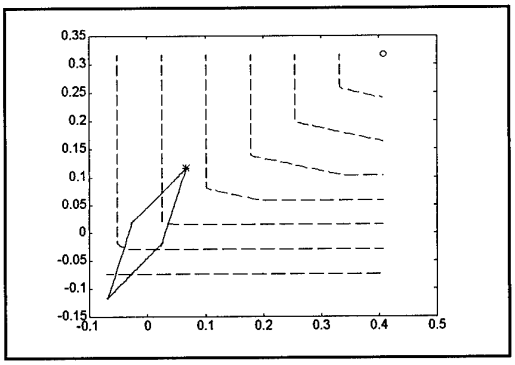


Figure 4.4 Feasibility Region (t=6)

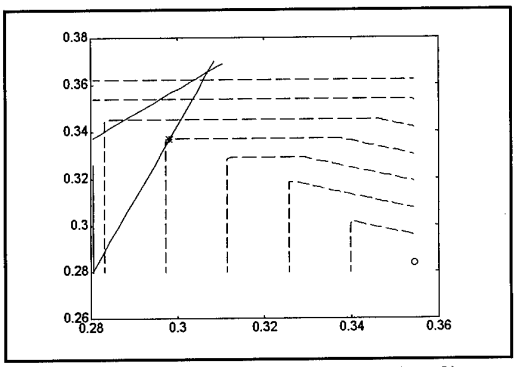


Figure 4.5 Feasibility Region (t=9)

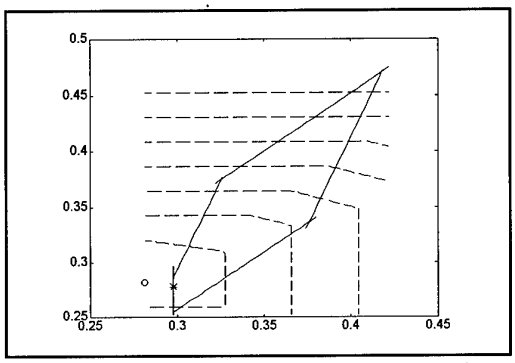


Figure 4.6 Feasibility Region (t=11)

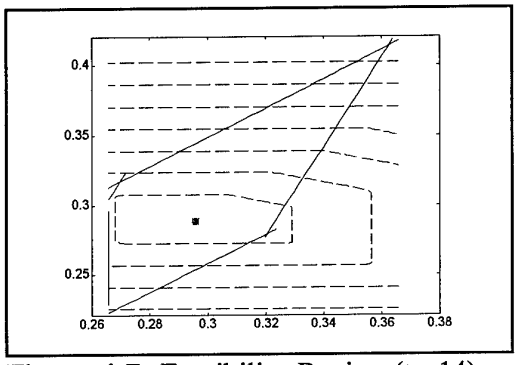


Figure 4.7 Feasibility Region (t=14)

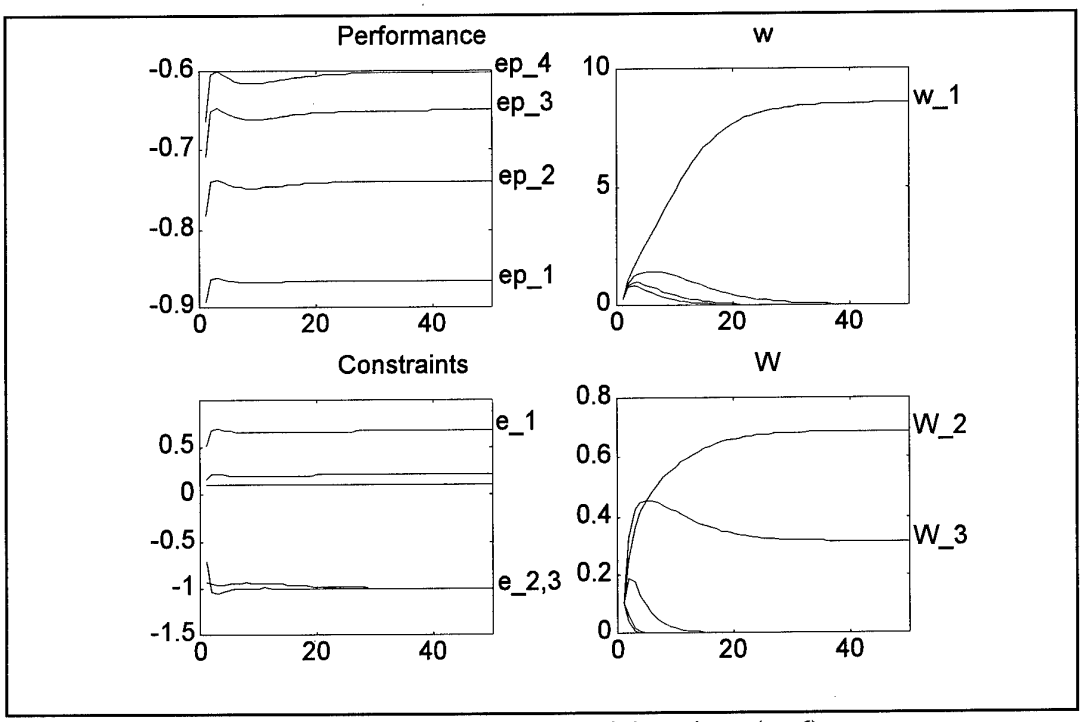


Figure 4.8 Example 4.2 - ℓ_∞-MWLS through 50 iterations (t=6)

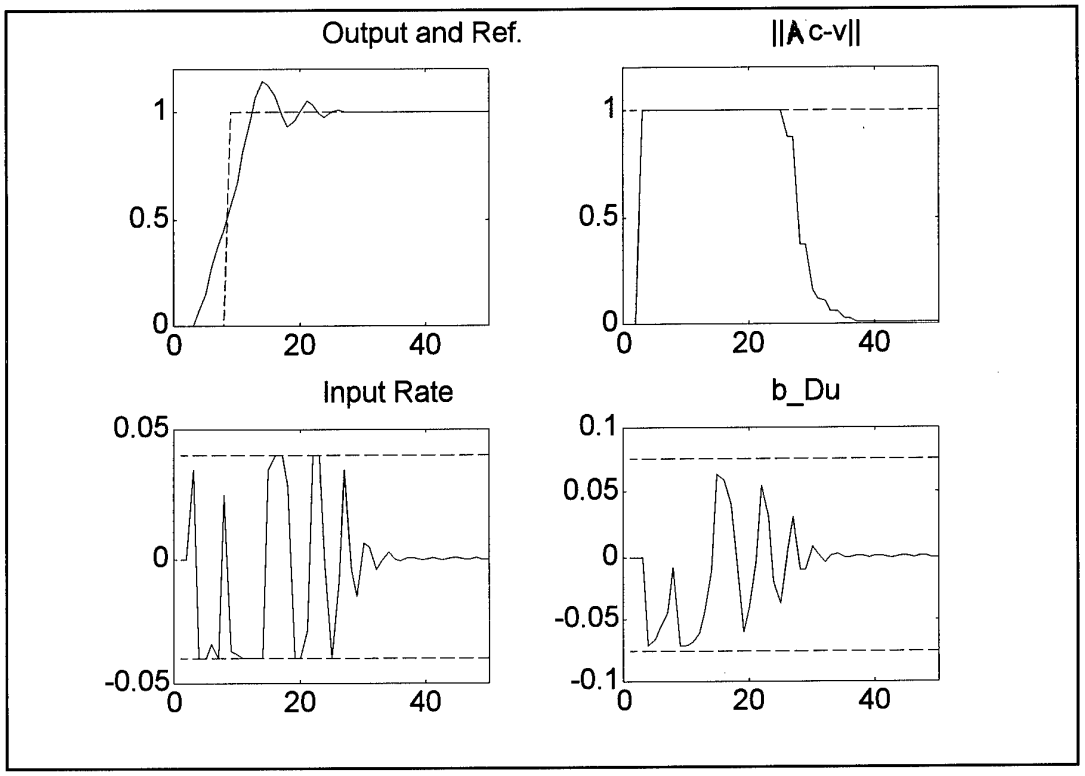


Figure 4.9 Example 4.3 - ℓ_∞-CSGPC response

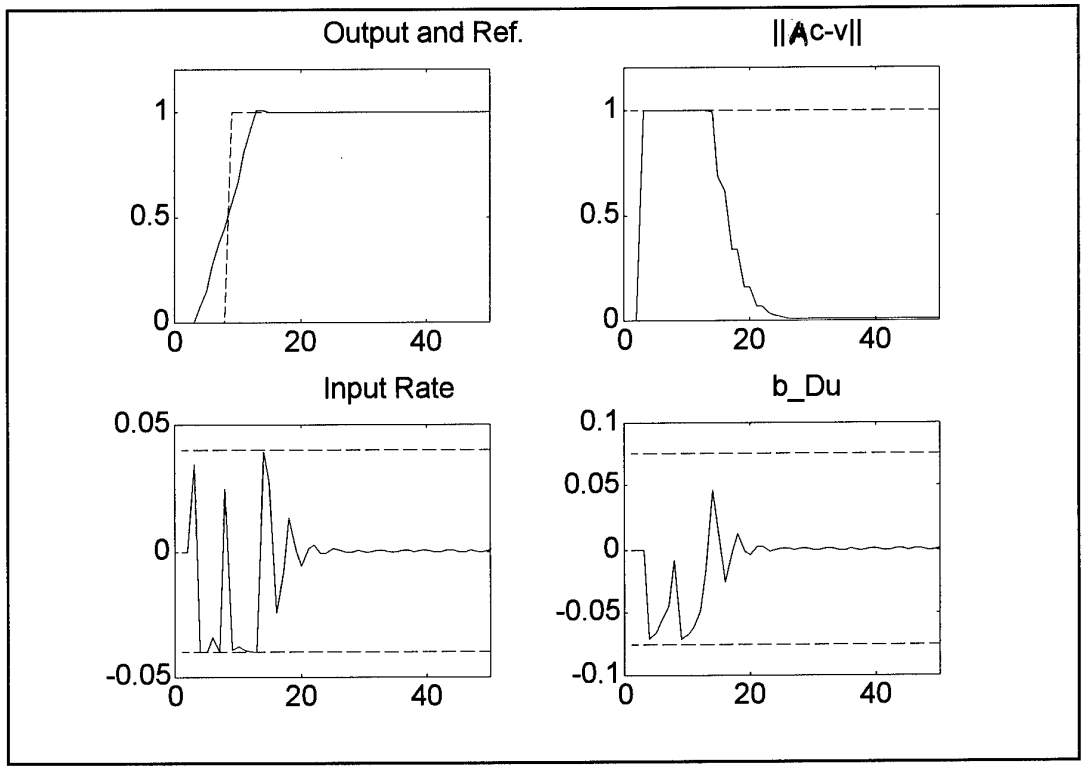


Figure 4.10 Example 4.3 - Mixed-Objective CSGPC response

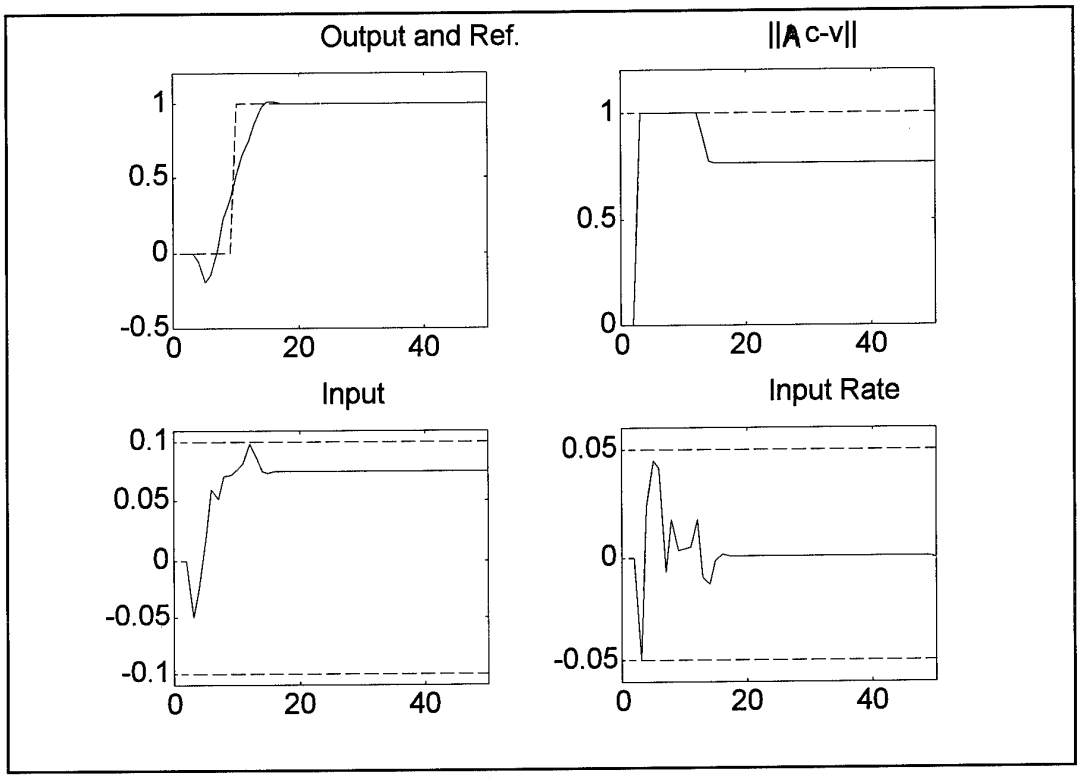


Figure 4.11 Example 4.4 - ℓ_∞-CSGPC response

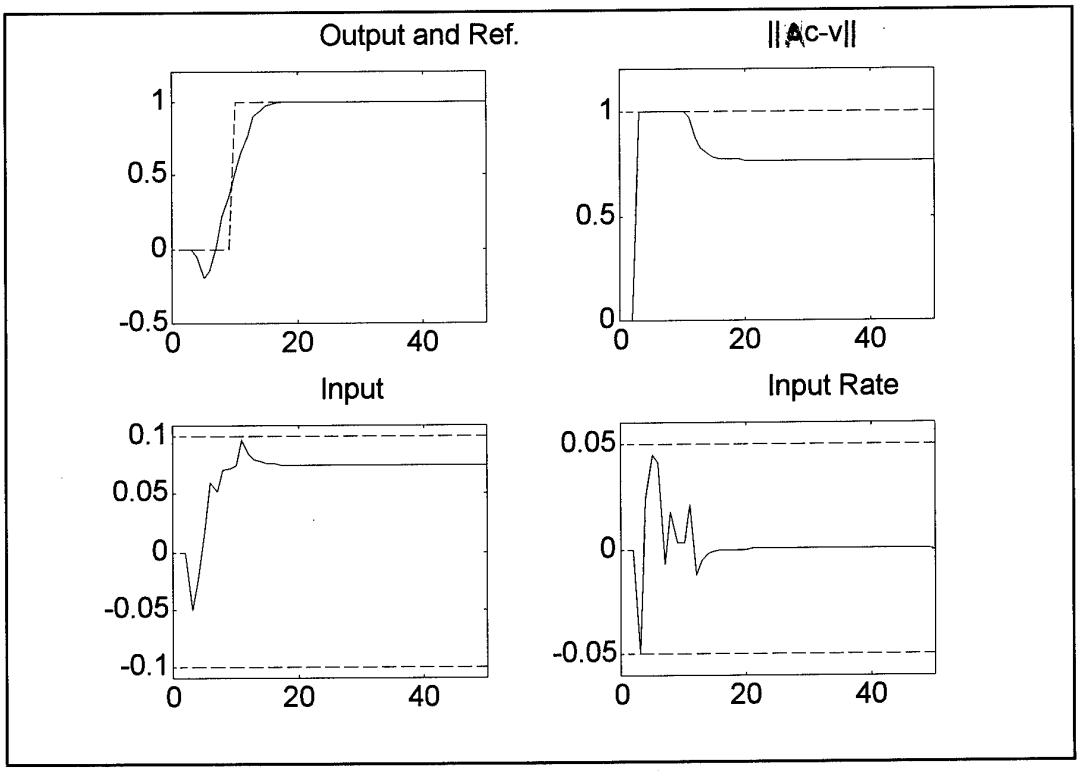


Figure 4.12 Example 4.4 - Mixed-Objective CSGPC response

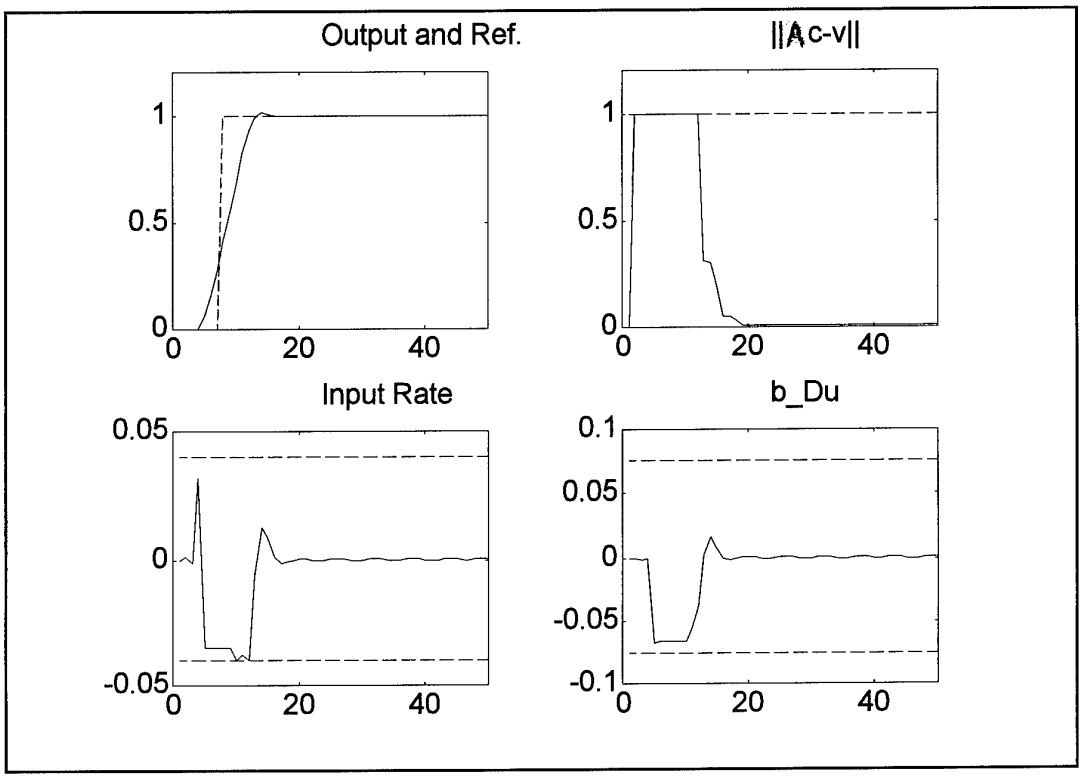


Figure 4.13 Example 4.5 - MCSGPC response

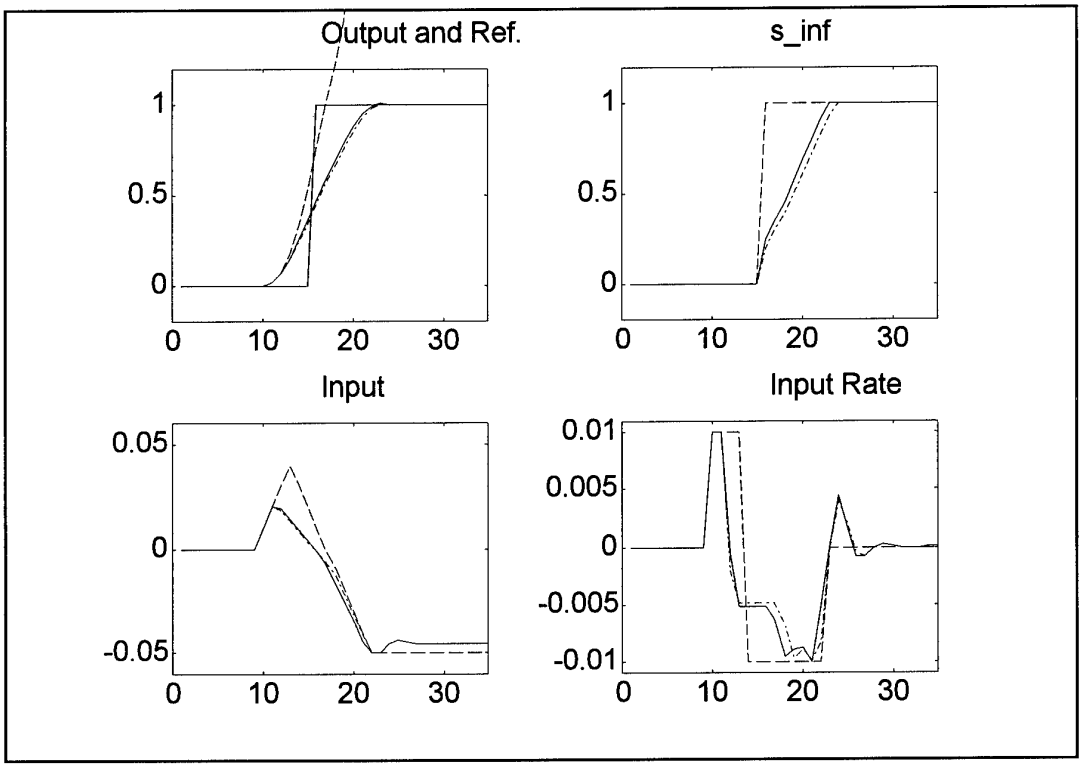


Figure 4.14 Example 4.6 - MCSGPC and RM response

Chapter 5

Cautious Stable Predictive Control

The stability result of SGPC may be viewed as having been obtained through the imposition of terminal constraints. The stabilizing loop of Figure 2.1, in conjunction with bezout identity (2.4), implicitly imposes dead-beat terminal constraints, namely that the predicted tracking errors should all be zero beyond a given output horizon and that the control moves themselves should become zero beyond a given input horizon. These terminal constraints may come into conflict with limits on system inputs (dictated by physical constraints); such a conflict, which we have termed STIF, may lead to instability. One possible remedy is to use longer horizons, but this results in a significant increase in the computation of constrained optima. An alternative is offered by set-point conditioning ([2], [14]), but inherent in this approach is a sacrifice of optimality in tracking in favour of retaining feasibility and hence guaranteeing stability. MCSGPC provides an alternative which does not sacrifice optimality, because it retains the actual set-point in the objective while basing the steady-state value of the output's end-point constraint on a slack variable; after imposition of the slack variable end-point constraint, the effect it to replace the terminal equality constraint with a terminal inequality constraint. In this chapter, we use a different approach to infeasibility: earlier terminal

constraints guarantee stability but actually use a sub-class of all stable input/output predictions, so here we employ constraints which are both necessary and sufficient for stability and thus, for a given number of degrees of freedom, maximize the control authority which is available for improving performance while respecting input constraints. The class of input/output predictions which are necessary and sufficient for stability will be those of Section 3.2.1.

A convenient way to guarantee the stability of predictive control strategies is to force the predicted trajectories of both output error and control increments to be finite length sequences (FLS). This can be effected by a process of "cancellation" of all the poles of the plant and its inverse, ie a pseudo cancellation (as per Remark 3.1) of all the poles and zeros. It has been recognised that for the purposes of stability one actually need only "cancel" the unstable poles ([30], [54]), thereby yielding predicted output error trajectories which are infinite length sequences (ILS). In Section 5.1, we show that it is also the case that one needs only "cancel" the nonminimum phase zeros with the effect of getting predicted control increment trajectories which are ILS; we also propose simple algorithms which implement these two changes in philosophy, but retain a finite length cost. In Section 5.2, an infinite horizon cost which is the sum of the square of the ILS errors and input increments is used, and two procedures are proposed for calculating this infinite length sum. Then, in Section 5.3, these results are applied to the problem of controlling plasma vertical position in the Compass-D tokamak test device. The resulting controller is compared to one produced with standard H_∞ design techniques. This application has been reported elsewhere ([47], [48]) and is briefly summarized here to demonstrate the applicability of the results of Sections 5.1 and 5.2.

For systems subject to physical constraints, ILS trajectories lead to a practical

difficulty; the physical constraints must be invoked over an infinite horizon. This problem can be overcome through the use of suitable input/output horizon bounds. A set of such bounds with respect to output constraints have been proposed elsewhere ([30], [54]). Here, we are concerned with input constraints only, but we explore the use of ILS predictions for both inputs and outputs; therefore, we require bounding results on inputs rather than outputs. In Section 5.4, we develop simple input bounding techniques which provide an efficient means of invoking the constraints over a infinite horizon by enforcing them over a finite horizon. Thus, we are able to use our necessary and sufficient terminal constraints to advantage by allowing for: i) the use of short command horizons; and/or ii) the release of control authority for better transient performance. We also consider the use of terminal inequality constraints similar those introduced in the previous chapter for MCSGPC and thus remove restrictions on the size of set-point changes. The efficacy/superiority of the resulting algorithm is illustrated by means of a numerical example.

5.1 Cautious Stable Control

The guarantee of stability afforded by SGPC (and other stable predictive algorithms) is based on the fact that, under the implied end-point constraints, the GPC type of performance index minimized by the algorithm forms a stable Lyapunov function. However, the end-point constraints in question, though sufficient for the proof of stability, are unnecessarily stringent and are responsible for the highly tuned nature of the resulting controllers. Highly tuned controllers may be desirable in some cases, but could lead to feasibility and stability problems in the case of plants subject to input

constraints. In Section 5.1.1, we relax the SGPC requirement by defining terminal constraints which are both necessary and sufficient for the guarantee of stability. The implication of this development is that rather than insist that both the predicted inputs and outputs should reach their required steady-state values within the given (finite) input/output horizons, now they are constrained to do so only in an asymptotic sense. As a consequence, the resulting controllers are less highly tuned than those given by SGPC and therefore possess improved robustness properties and are more suitable for use in the control of systems which are subject to physical constraints.

Using the SGPC framework, in Section 5.1.2, we introduce further predictive control algorithms with guaranteed stability: cautious mean-level (CaML), and cautious stable predictive control (CaSC). These algorithms progressively relax the stringent end-point constraints deployed in SGPC: in particular, CaML does away with the requirement that the predicted output should reach its desired value within the given output horizon, and CaSC further removes the requirement that the predicted control increment should be zero beyond the given control horizon. In fact, CaSC is based on conditions which, given a fixed number of degrees of freedom, are both necessary and sufficient for the guarantee of stability, and so any further relaxation of the SGPC constraints is not possible without increasing the number of control degrees of freedom. Concomitant with the constraint relaxation is a reduction in the level of control activity and the derivation of less highly tuned and more robust predictive controllers; the robustness properties of CaSC will be investigated in Section 5.1.3. The efficacy of the new algorithms, their superiority in dealing with physical constraints, and their improved robustness properties are illustrated by means of design studies in Section 5.1.4.

5.1.1 Stability through terminal constraints

The important elements in "Lyapunov" stability arguments (eg. Theorem 2.1) for any stable predictive control algorithm are i) that the proposed cost is finite over an infinite horizon and ii) that the optimal control law used at t can always be used again at $t+1$. With these two elements, it is easy to show that the cost is a monotonically decreasing function of time, namely, that it constitutes a stable Lyapunov function. In the absence of input constraints, ii) is always possible; systems subject to input constraints will be the topic of Section 5.4. To achieve i), the predictions which go into the cost must either be of finite duration (a FLS), or must asymptotically approach zero (an ILS); this is achieved (either explicitly or implicitly) through the use of terminal constraints. In SGPC, the stabilizing loop forces the output errors and input increments to be FLS's and thus implicitly imposes the terminal constraints that the predicted output error should be zero after n_y steps and that the predicted input increment should be zero after n_u steps:

$$r_{t+n_r}-y_{t+i}=0, \quad \forall\ i>n_y; \qquad \Delta u_{t+i}=0, \quad \forall\ i\geq n_u \tag{5.1}$$

These same constraints are imposed explicitly in [8] and [27] through the use of equality constraints; the advantage of SGPC is an explicit expression for the degrees of freedom (DOF) remaining after imposition of the terminal constraints. Other work recognized that these constraints are overly restrictive, in that the output errors need only approach zero asymptotically [30]; by requiring that the unstable modes be zero after the output horizon and utilizing a finite number of control changes, the algorithm of Rawlings and Muske (which we will term RM) explicitly imposes these terminal constraints:

$$r_{t+n_r}-y_{t+i}=0, \quad i\to\infty; \qquad \Delta u_{t+i}=0, \quad \forall\ i\geq n_u \tag{5.2}$$

Thus, the output errors form an ILS, but the input increments are still a FLS. These constraints are also restrictive in that the predicted control increments need only approach

zero asymptotically; thus, the least restrictive set of terminal constraints which allows a Lyapunov stability proof utilize ILS for both the output errors and the input increments:

$$r_{t+n_r} - y_{t+i} = 0, \quad i \to \infty; \qquad \Delta u_{t+i} = 0, \quad i \to \infty \tag{5.3}$$

The prediction equations of Section 3.2.1, which were derived as necessary and sufficient conditions for stable predictions, implicitly impose these terminal constraints and thus utilize the necessary and sufficient conditions for imposition of the Lyapunov stability arguments. An implementable algorithm (at least when physical constraints are considered) requires a finite number of DOF over which to optimize; because the DOF remaining after imposition of constraints (5.3) are expressed explicitly in Section 3.2.1 as the current and future values of c, these future values can be taken to be a FLS and thus can be used for optimization rather than the ILS of future control increments. First, we repeat the derivation of the class of stable prediction equations, but in a more convenient form involving the future errors, $e=r-y$.

Let $\underset{\rightarrow}{\Delta r}=[r_{t+1}, (r_{t+2}-r_{t+1}), (r_{t+2}-r_{t+1}), \ldots, (r_{t+n_r}-r_{t+n_r-1})]^T$ be the vector of future set-point increments, where n_r is the reference horizon and r is assumed to be constant thereafter. Furthermore, let $\Delta r(z)$ be a polynomial whose coefficients are the elements of $\underset{\rightarrow}{\Delta r}$, such that $r(z)=\Delta r(z)/\Delta(z)$. Then, the z-transform of the future errors is obtained by subtracting $y(z)$ (eqn. (3.31)b) from $r(z)$; with a common denominator, we have:

$$e(z)=r(z)-y(z)=\frac{a(z)\Delta r(z)-b(z)\Delta u(z)-p(z)}{a^-(z)\alpha^+(z)}=\frac{q(z)-b(z)\Delta u(z)}{a^-(z)\alpha^+(z)}; \qquad q(z)=a(z)\Delta r(z)-p(z) \tag{5.4}$$

Then, the necessary and sufficient condition for the stability of the predicted error, e, is that the numerator of eqn. (5.4)a contains as a factor the unstable system poles:

$$q(z)-b(z)\Delta u(z)=\alpha^+(z)\phi(z) \qquad \text{or} \qquad \Delta u(z)=-\frac{\alpha^+(z)\phi(z)-q(z)}{b^-(z)b^+(z)} \tag{5.5}$$

where $\phi(z)$ is the z-transform of a convergent sequence $\{\phi_0, \phi_1, \phi_2, \ldots\}$. It follows that the

predicted trajectories of control increments, Δu, will be stable if, and only if, the numerator of eqn. (5.5)b contains the "unstable" system zeros:

$$\alpha^{+}(z)\phi(z)-q(z)=b^{+}(z)\psi(z) \qquad \text{or} \qquad \alpha^{+}(z)\phi(z)-b^{+}(z)\psi(z)=q(z) \tag{5.6}$$

where $\psi(z)$ is a polynomial with a convergent sequence of coefficients. Diophantine eqn. (5.6)b has the same LHS as eqn. (3.33), but a different RHS, implying a different particular solution. Thus, with $n_{\phi}=\max[n_{b^{+}}-1, n_{a^{-}}+n_r-2]$, $n_{\psi}=n_{a^{+}}$, a minimal solution of diophantine eqn. (5.6)b is:

$$\begin{bmatrix}\phi_p\\ \psi_p\end{bmatrix}=\begin{bmatrix}\Gamma_{\alpha^{+}} & -\Gamma_{b^{+}}\end{bmatrix}^{-1}\begin{bmatrix}v_q\\ \mathbf{0}\end{bmatrix}=\begin{bmatrix}P_1 & P_3\\ P_2 & P_4\end{bmatrix}\begin{bmatrix}v_q\\ \mathbf{0}\end{bmatrix}=\begin{bmatrix}P_1\\ P_2\end{bmatrix}v_q \tag{5.7}$$

where $P_{1\text{-}4}$ are as defined for eqn. (3.35)b, and the elements of v_q are the coefficients of $q(z)$; from eqns. (5.4)b and (3.31)c, v_q is given as:

$$v_q=\Gamma_a \underset{\rightarrow}{\Delta r}+H_{\alpha}\underset{\leftarrow}{y}-H_b\underset{\leftarrow}{\Delta u}=[\Gamma_{\alpha}\ \theta_{\alpha}]\underset{\rightarrow}{r}+H_{\alpha}\underset{\leftarrow}{y}-H_b\underset{\leftarrow}{\Delta u} \tag{5.8}$$

Then, the general solution to (5.6)b is:

$$\phi(z)=b^{+}(z)c(z)+\phi_p(z); \qquad \psi(z)=\alpha^{+}(z)c(z)+\psi_p(z) \tag{5.9}$$

where $c(z)$ is a polynomial with a sequence of coefficients which converges to zero; it contains all the available degrees of freedom. Inserting (5.9)a,(5.9)b into (5.5)a,(5.6)a and then into (5.4)a,(5.5)b gives:

$$e(z)=\frac{b^{+}(z)c(z)+\phi_p(z)}{a^{-}(z)}; \qquad \Delta u(z)=-\frac{\alpha^{+}(z)c(z)+\psi_p(z)}{b^{-}(z)} \tag{5.10}$$

Theorem 5.1 The entire class of stable error/output prediction equations for the plant, $z^{-1}b(z)/a(z)$, is given by eqns. (5.10); furthermore, these equations represent the class of predictions which satisfy terminal constraints (5.3).

Proof: This is true by derivation. □

5.1.2 Simple CaSC and CaML algorithm

A convenient way to prove stability is to define a suitable cost which uses (FLS) predictions; the conditions of Section 5.1.1 take us part of the way towards FLS because of the implicit pseudo cancellation of $\alpha^+(z)$ (in eqn. (5.4)a) and $b^+(z)$ (in eqn. (5.5)b). It remains therefore to: (i) "cancel" $a^-(z)$ (in eqn. (5.10)a) and $b^-(z)$ (in eqn. (5.10)b); (ii) invoke an appropriate terminal constraint; and (iii) define a cost which can be shown to be monotonically decreasing. Now (i) implies the definition of new prediction variables $\tilde{e}_t = a^-(z)e_t$, $\Delta\hat{u}_t = b^-(z)\Delta u_t$; (ii) implies that future values of $\tilde{e}$ must be zero after some output horizon n_y, and that the future values of $\Delta\hat{u}$ must be zero after some input horizon n_u; (iii) implies that J must be based on $\tilde{e}$ and $\Delta\hat{u}$ rather than e and Δu. Collecting these observations together, we therefore write:

$$\tilde{e}(z) = \sum_{i=1}^{n_y} \tilde{e}_{t+i} z^{-i+1}; \qquad \Delta\hat{u}(z) = \sum_{i=0}^{n_u-1} \Delta\hat{u}_{t+i} z^{-i}$$
$$\tilde{J}_{CaSC} = \sum_{i=1}^{n_y} \tilde{e}_{t+i}^2 + \lambda \sum_{i=0}^{n_u-1} \Delta\hat{u}_{t+i}^2 = \| \underset{\rightarrow}{\tilde{e}} \|_2^2 + \lambda \| \underset{\rightarrow}{\Delta\hat{u}} \|_2^2 \qquad \textbf{(5.11)}$$

$\underset{\rightarrow}{\tilde{e}}$ and $\underset{\rightarrow}{\Delta\hat{u}}$ are vectors of the coefficients of $\tilde{e}(z)$, $\Delta\hat{u}(z)$. Implicitly, we have invoked the following terminal constraints:

$$\tilde{e}_{t+i} = 0, \quad \forall\ i > n_y; \qquad \Delta\hat{u}_{t+i} = 0, \quad \forall\ i \geq n_u \qquad \textbf{(5.12)}$$

Because $a^-(z)$ and $b^-(z)$ are stable, these terminal constraints can be seen to imply those of (5.3), but they also imply a finite number of degrees of freedom; this is precisely what is needed for an implementable constrained optimization.

For terminal constraints (5.12) to be satisfied, the coefficients of the numerators of eqns. (5.10) must be FLS, and the degree of these polynomials must be n_y-1 and n_u-1. So choosing $c(z) = c_0 + c_1 z^{-1} + \cdots + c_{n_c-1} z^{-n_c+1}$, we can write the vector forms of the prediction equations as:

$$\begin{aligned} \underset{\rightarrow}{e} &= \Gamma_{1/a}\text{-}(\Gamma_{b}\text{-}\underset{\rightarrow}{c} + P_1 v_q); & \underset{\rightarrow}{\tilde{e}} &= \Gamma_{b}\text{-}\underset{\rightarrow}{c} + P_1 v_q + H_a\text{-}\underset{\leftarrow}{e} \\ \underset{\rightarrow}{\Delta u} &= -\Gamma_{1/b}\text{-}(\Gamma_{\alpha}\text{-}\underset{\rightarrow}{c} + P_2 v_q); & \underset{\rightarrow}{\Delta\hat{u}} &= -\Gamma_{\alpha}\text{-}\underset{\rightarrow}{c} - P_2 v_q + H_b\text{-}\underset{\leftarrow}{\Delta u} \end{aligned} \qquad \textbf{(5.13)}$$

where, as before, the i^{th} column of the Γ matrices contains the impulse response of the indicated transfer function multiplied by z^{-i+1}. Eqns. (5.13)a,c are prediction equation forms of eqns. (5.10), and eqns. (5.13)b,d are derived from $\tilde{e}_t = a^{-}(z)\, e_t, \Delta\hat{u}_t = b^{-}(z)\,\Delta u_t$ which imply the prediction relationships: $\underset{\rightarrow}{\tilde{e}} = C_a\text{-}\underset{\rightarrow}{e} + H_a\text{-}\underset{\leftarrow}{e}$; $\underset{\rightarrow}{\Delta\hat{u}} = C_b\text{-}\underset{\rightarrow}{\Delta u} + H_b\text{-}\underset{\leftarrow}{\Delta u}$. Terminal constraints (5.12) are satisfied with prediction equations (5.13)b,d where n_y and n_u are related to the command and/or reference horizons, n_c, n_r, by $n_y = \max[n_c + n_b\text{-}, n_r + n_a\text{-} - 1]$, and $n_u = \max[n_c + n_a\text{-} + 1, n_b\text{-}]$.

Remark 5.1 In the strictest sense, necessary and sufficient conditions for stable predictions would necessitate that $n_c \to \infty$, or to put it another way, while terminal constraints (5.12) imply those of (5.3), terminal constraints (5.3) imply those of (5.12) only for n_c infinite. This is impractical, and indeed, for reducing computational complexity, n_c should be chosen to be small. In the rest of the chapter, the term, necessary and sufficient, will include the assumption that the number of degrees of freedom over which the performance can be optimized is finite and equal to n_c.

Prediction equations (5.13)b,d imply that the cost $\tilde{J}_{CaSC}$ of (5.11) is quadratic in the vector of future c's and hence (in the absence of physical system constraints) can be minimized explicitly. The optimal vector of c's, through eqn. (5.13)c, defines the optimal vector of future control increments, the first of which is implemented. This defines the prediction and optimization cycle of CaSC which is repeated at each time instant. Because $\tilde{e}$, $\Delta\hat{u}$ are FLS, $\tilde{J}_{CaSC}$ is of the same form as J in SGPC, thus it is easy

to show that $\tilde{J}_{CaSC}$ is a stable Lyapunov function and therefore that CaSC has guaranteed stability and asymptotic tracking.

CaSC uses necessary and sufficient conditions for stable predictions and hence has more control authority available for handling constraints and performance than SGPC. However, it results in ILS for both output and input predictions; therefore ensuring that input constraints are met may necessitate the use of long constraint horizons, thereby increasing significantly the computation burden. In cases like this, it may be advantageous to sacrifice some of the extra degrees of freedom generated by CaSC in order to obtain FLS for the input predictions; this is accomplished by utilizing terminal constraints (5.2).

These are obtained by reassigning $b^+(z)=b(z)$ and $b^-(z)=1$ in the development of Section 5.1.1, so that the predicted errors form an ILS and the predicted input increments form a FLS. Obviously, $\Delta\hat{u}(z)=\Delta u(z)$, and thus the cost (which is again a stable Lyapunov function) is:

$$\tilde{J}_{CaML}=\sum_{i=1}^{n_y}(\tilde{r}_{t+i}-\tilde{y}_{t+i})^2+\lambda\sum_{i=0}^{n_u-1}\Delta u_{t+i}^2 = \left\|\vec{\tilde{r}}-\vec{\tilde{y}}\right\|_2^2+\lambda\left\|\vec{\Delta u}\right\|_2^2 \qquad \textbf{(5.14)}$$

We note that the earlier algorithm (Mean Level Predictive Control) presented in [36] is a special case of that presented here, derived for $\lambda=\infty$. CaML also bears a similarity to RM and the algorithm of [43], in that they all cancel the unstable poles in the output predictions; however the indices of performance are different.

Remark 5.2 It can be shown that the effect of weighting the predicted errors by $a^-(z)$ (and the predicted input increments by $b^-(z)$ for CaSC) is to cause $a^-(z)$ (and $b^-(z)$ for

CaSC) to be a factor of the closed-loop pole polynomial, $p_c(z)$. If any of the roots of these polynomials are slow, the effect may be undesirable. This problem is easily remedied by redefining $a^+(z)$ and/or $b^+(z)$ to include the relevant slow roots, eg. all roots with modulus greater than or equal to ρ ($\rho \leq 1$). Such a modification would ensure that slow roots would be excluded from the closed-loop dynamics, and in the limit (for $\rho=0$), would yield SGPC.

5.1.3 Robustness analysis and optimization for CaSC

The robustness analysis for CaSC follows along the same lines as the analysis for SGPC in Section 2.2. Introducing prediction eqns. (5.13)c,d into $\tilde{J}_{CaSC}$ gives:

$$\begin{aligned}\tilde{J}_{CaSC} &= (\Gamma_{b^*}\underset{\rightarrow}{\boldsymbol{c}} + P_1 \boldsymbol{v}_q + H_{a^-}\underset{\leftarrow}{\boldsymbol{e}})^T(\Gamma_{b^*}\underset{\rightarrow}{\boldsymbol{c}} + P_1 \boldsymbol{v}_q + H_{a^-}\underset{\leftarrow}{\boldsymbol{e}}) + \lambda(\Gamma_{\alpha^*}\underset{\rightarrow}{\boldsymbol{c}} + P_2 \boldsymbol{v}_q - H_{b^-}\underset{\leftarrow}{\Delta \boldsymbol{u}})^T(\Gamma_{\alpha^*}\underset{\rightarrow}{\boldsymbol{c}} + P_2 \boldsymbol{v}_q - H_{b^-}\underset{\leftarrow}{\Delta \boldsymbol{u}}) \\ &= \underset{\rightarrow}{\boldsymbol{c}}^T(\Gamma_{b^*}^T\Gamma_{b^*} + \lambda\Gamma_{\alpha^*}^T\Gamma_{\alpha^*})\underset{\rightarrow}{\boldsymbol{c}} + \underset{\rightarrow}{\boldsymbol{c}}^T(\Gamma_{b^*}^T(P_1 \boldsymbol{v}_q + H_{a^-}\underset{\leftarrow}{\boldsymbol{e}}) + \lambda\Gamma_{\alpha^*}^T(P_2 \boldsymbol{v}_q - H_{b^-}\underset{\leftarrow}{\Delta \boldsymbol{u}})) + \gamma \qquad (5.15)\end{aligned}$$

$$\gamma = (P_1 \boldsymbol{v}_q + H_{a^-}\underset{\leftarrow}{\boldsymbol{e}})^T(P_1 \boldsymbol{v}_q + H_{a^-}\underset{\leftarrow}{\boldsymbol{e}}) + \lambda(P_2 \boldsymbol{v}_q - H_{b^-}\underset{\leftarrow}{\Delta \boldsymbol{u}})^T(P_2 \boldsymbol{v}_q - H_{b^-}\underset{\leftarrow}{\Delta \boldsymbol{u}})$$

which is minimized with respect to $\underset{\rightarrow}{\boldsymbol{c}}$ by:

$$\underset{\rightarrow}{\boldsymbol{c}_o} = -(\Gamma_{b^*}^T\Gamma_{b^*} + \lambda\Gamma_{\alpha^*}^T\Gamma_{\alpha^*})^{-1}(\Gamma_{b^*}^T(P_1 \boldsymbol{v}_q + H_{a^-}\underset{\leftarrow}{\boldsymbol{e}}) + \lambda\Gamma_{\alpha^*}^T(P_2 \boldsymbol{v}_q - H_{b^-}\underset{\leftarrow}{\Delta \boldsymbol{u}})) \qquad (5.16)$$

Using the definition of $\boldsymbol{v}_q$ in eqn. (5.8), the first optimal element, c_t, is given as:

$$c_t = -\underset{\rightarrow}{\boldsymbol{p}_{ra}}\underset{\rightarrow}{\boldsymbol{r}} - \underset{\leftarrow}{\boldsymbol{p}_{rc}}\underset{\rightarrow}{\boldsymbol{r}} - \underset{\leftarrow}{\boldsymbol{p}_y}\underset{\leftarrow}{\boldsymbol{y}} + \underset{\leftarrow}{\boldsymbol{p}_u}\underset{\leftarrow}{\Delta \boldsymbol{u}}$$

$$\begin{aligned}\underset{\rightarrow}{\boldsymbol{p}_{ra}} &= \boldsymbol{p}^T(\Gamma_{b^*}^T P_1 + \lambda\Gamma_{\alpha^*}^T P_2)[\Gamma_\alpha \ \boldsymbol{\theta}_\alpha]; & \underset{\leftarrow}{\boldsymbol{p}_{rc}} &= \boldsymbol{p}^T\Gamma_{b^*}^T H_{a^-} \\ \underset{\leftarrow}{\boldsymbol{p}_y} &= \boldsymbol{p}^T(\Gamma_{b^*}^T(P_1 H_\alpha - H_{a^-}) + \lambda\Gamma_{\alpha^*}^T P_2 H_\alpha) & \boldsymbol{p}^T &= \boldsymbol{e}_1^T(\Gamma_{b^*}^T\Gamma_{b^*} + \lambda\Gamma_{\alpha^*}^T\Gamma_{\alpha^*})^{-1} \\ \underset{\leftarrow}{\boldsymbol{p}_u} &= \boldsymbol{p}^T(\Gamma_{b^*}^T P_1 H_b + \lambda\Gamma_{\alpha^*}^T(P_2 H_b - H_{b^-})) \end{aligned} \qquad (5.17)$$

and, from eqn. (5.13)c, the first optimal input increment, Δu_t, is:

$$\Delta u_t = -c_t - \boldsymbol{e}_1^T P_2([\Gamma_\alpha \ \boldsymbol{\theta}_\alpha]\underset{\rightarrow}{\boldsymbol{r}} + H_\alpha \underset{\leftarrow}{\boldsymbol{y}} - H_b \underset{\leftarrow}{\Delta \boldsymbol{u}}) \qquad (5.18)$$

Next, we define the vectors:

$$\boldsymbol{D}_k=[1 \quad (\boldsymbol{p}_u - \boldsymbol{e}_1^T P_2 H_b)]; \qquad \boldsymbol{N}_k=\boldsymbol{e}_1^T P_2 H_\alpha - \boldsymbol{p}_y; \qquad \boldsymbol{p}_{ra}'=\boldsymbol{p}_{ra} - \boldsymbol{e}_1^T P_2 [\Gamma_\alpha \ \theta_\alpha] \tag{5.19}$$

Let the coefficients of the polynomials, $D_k(z)$, $N_k(z)$, be the elements of the vectors, $\boldsymbol{D}_k$, $\boldsymbol{N}_k$; and let the coefficients of the bi-causal polynomial, $p_r(z)$, be the elements of the vectors, $\boldsymbol{p}_{rc}$ and $\boldsymbol{p}_{ra}'$, such that, if $\boldsymbol{p}_{ra}=[p_{ra1},p_{ra2},\cdots,p_{ran_a}]$ and $\boldsymbol{p}_{rc}=[p_{rc1},p_{rc2},\cdots,p_{rcn_r}]$, then:

$$p_r(z)=p_{rcn_a}z^{-n_a}+\cdots+p_{rc2}z^{-2}+p_{rc1}z^{-1}+p_{ra1}+p_{ra2}z^{1}+\cdots+p_{ran_r}z^{n_r-1} \tag{5.20}$$

Combining eqns. (5.17)a and (5.18) and the preceding definitions, we get $D_k(z)\Delta u_t=p_r(z)r_{t+1}-N_k(z)y_t$ which is the same as eqn. (2.22)b, and so CaSC can be implemented with the feedback configuration of Figure 2.2. Thus, as with SGPC, there is an entire class of optimal controllers, defined by $D(z)=D_k(z)-z^{-1}b(z)Q(z)$ and $N(z)=N_k(z)+\alpha(z)Q(z)$, where $Q(z)$ is an arbitrary stable polynomial, which can be used to improve properties such as robustness and/or noise handling *without* affecting performance. The arguments and procedures are identical to those given in Section 2.2.

5.1.4 Simulation results and comparisons

The motivation behind the proposed stable predictive control algorithms is the use of less conservative (and for CaSC, necessary and sufficient) conditions for the stability of input/output predictions. The resulting classes of predictions are less restrictive and therefore wider than that of SGPC, and thus provide the means of avoiding overactive input/output responses. This point will be illustrated here by means of numerical examples which study the closed-loop responses to step changes in the set-point r as produced by each of the three algorithms, SGPC, CaML, and CaSC. For completeness, we also give the corresponding simulation plots for GPC. As expected, it will be seen that SGPC drives the system hard and hence uses highly active inputs; this is because the

transferences from c to u and c to y are FLSs. For CaML, only the transfer function from c to u is a FLS, so the inputs are less active. The transferences in CaSC are all ILSs, and hence it results in low input activity while demonstrating reasonable performance. GPC, because it uses finite input horizons, but no end-point constraints, typically falls between CaML and CaSC, though of course, it has no stability guarantee.

The examples provide a comparison of the robustness properties of SGPC, CaML, GPC, and CaSC and demonstrate the superiority of CaSC, both for $Q(z)=0$ and for the respective choices of optimal fourth order $Q(z)$. In addition, the second example is used to illustrate the benefits of redefining $a^+(z)$ and $b^+(z)$ to include stable, but "slow" poles/zeros, as per Remark 5.2.

Example 5.1 Let the model be given by $a(z)=1\text{-}1.6z^{-1}+0.13z^{-2}+0.21z^{-3}$ and $b(z)=1\text{-}2.7z^{-1}+1.4z^{-2}$, which has an unstable pole at $z=1.4$ and a non-minimum phase zero at $z=2$. Furthermore, let the control parameters be: $\lambda=1$, $n_r=3$, and $n_c=2$ for SGPC, CaML, and CaSC; and $n_u=2$ and $n_y=6$ for GPC. In this example and the following, SGPC results will be indicated with dotted lines, CaML with dashed lines, GPC with dash-dotted lines, and CaSC with solid lines. The output/input responses are shown in Figure 5.1a,b and illustrate the expected characteristic: SGPC is the most highly tuned and has very active input/output responses; CaSC is the least active and gives good performance (in fact, it settles as quickly as SGPC).

The nominal ($Q(z)=0$) robustness indicator (for $W(z)=1$), given by the modulus of the transfer function $K(z)S(z)$ of eqn. (2.25), is plotted in Figure 5.1c as a function of ωT, where $z=e^{j\omega T}$, $0\leq\omega T\leq\pi$. The expected ordering is demonstrated with CaSC being the most robust and SGPC the least. The robustness plots for the optimal 4th order $Q(z)$

are given in Figure 5.1d. Clearly, $Q(z)$ improves the robustness properties of all four algorithms, but as expected, does not alter their ranking.

Example 5.2 Let the model be given by $a(z)=1-1z^{-1}+0.01z^{-2}+0.12z^{-3}$ and $b(z)=1-2.4z^{-1}+0.8z^{-2}$, which has a non-minimum phase zero at $z=2$ and no unstable poles; however, it has a slow pole at $z=0.8$. Furthermore, let the control parameters be: $\lambda=1$, $n_r=1$, and $n_c=2$ for SGPC, CaML, and CaSC; and $n_u=2$ and $n_y=6$ for GPC. In this case, CaML and CaSC are so highly de-tuned that the output responses (Figure 5.2) are unsatisfactory. The reason for the slow responses can be traced to the open-loop pole at $z=0.8$, which automatically appears as a closed-loop pole. However, as mentioned in Remark 5.2, $a^+(z)$ can be redefined to include all roots on or outside an circle centred at the origin and of radius ρ; for the current example, a judicious choice for ρ is 0.75. The resulting responses (Figure 5.3a,b) are now satisfactory. Once again, CaSC results in input/output trajectories which are significantly less active than those of the other algorithms: i) the non-minimum phase behaviour during the immediate transients is less than half as pronounced for CaSC than it is for SGPC; ii) the maximum control amplitude for CaSC is 0.35, as compared to 0.55 for SGPC.

The robustness plots for $Q(z)=0$ and for the optimal 4^{th} order $Q(z)$ are given in Figure 5.3c,d and illustrate the expected ranking of the four algorithms: CaSC is the most robust and SGPC is the least.

5.2 Infinite Horizon Stable Predictive Control

The algorithms of the previous section use necessary and sufficient conditions for stable predictions, but the proposed costs penalise filtered rather than actual error/input increment predictions so as to convert infinite horizons into finite horizons; hence, while the algorithms are simple to implement, this simplicity could come at the expense of slow closed-loop poles (as per Remark 5.2). While this expense can be avoided by including the slow roots of $a(z)$, $b(z)$ in $a^+(z)$, $b^+(z)$ and thus moving toward SGPC, here we explore an alternative which penalizes the actual ILS error/input increment predictions.

Control laws have been implemented which use ILS outputs with FLS inputs [30], but in this section, we use ILS input and outputs and consider two methods of implementing the implied control law. The first method is largely an extension of the ideas presented in [30], and the second is an alternative which avoids the need for the solution of a Lyapunov equation and thus, for high order models, can be more efficient. The results of the section are illustrated by means of numerical examples which highlight the advantages of the proposed control algorithm.

5.2.1 Nominal stable control law

The Infinite Horizon Stable Predictive Control (IHSPC) cost is given as:

$$J_{CaSC}=\sum_{i=1}^{\infty}(r_{t+i}-y_{t+i})^2+\lambda\sum_{i=0}^{\infty}\Delta u_{t+i}^2 = \|\underset{\rightarrow}{r}-\underset{\rightarrow}{y}\|_2^2+\lambda\|\underset{\rightarrow}{\Delta u}\|_2^2 \qquad (5.21)$$

Stability is guaranteed by virtue of the infinite horizon deployed in performance index (5.21) which allows standard Lyapunov arguments to be used. Introducing prediction eqns. (5.13)a,c into J_{CaSC} gives:

$$J_{CaSC}=(\Gamma_{b\cdot}\underset{\rightarrow}{c}+P_1 v_q)^T\Gamma_{1/a}^T\Gamma_{1/a\cdot}(\Gamma_{b\cdot}\underset{\rightarrow}{c}+P_1 v_q)+\lambda(\Gamma_{\alpha\cdot}\underset{\rightarrow}{c}+P_2 v_q)^T\Gamma_{1/b}^T\Gamma_{1/b\cdot}(\Gamma_{\alpha\cdot}\underset{\rightarrow}{c}+P_2 v_q)$$
$$=(\Gamma_{b\cdot}\underset{\rightarrow}{c}+P_1 v_q)^T B_a(\Gamma_{b\cdot}\underset{\rightarrow}{c}+P_1 v_q)+\lambda(\Gamma_{\alpha\cdot}\underset{\rightarrow}{c}+P_2 v_q)^T B_b(\Gamma_{\alpha\cdot}\underset{\rightarrow}{c}+P_2 v_q) \quad \textbf{(5.22)}$$

which is minimized with respect to $\underset{\rightarrow}{c}$ by:

$$\underset{\rightarrow}{c_o}=-(\Gamma_{b\cdot}^T B_a\Gamma_{b\cdot}+\lambda\Gamma_{\alpha\cdot}^T B_b\Gamma_{\alpha\cdot})^{-1}(\Gamma_{b\cdot}^T B_a P_1+\lambda\Gamma_{\alpha\cdot}^T B_b P_2)v_q \quad \textbf{(5.23)}$$
$$B_a=\Gamma_{1/a}^T\Gamma_{1/a\cdot} \qquad B_b=\Gamma_{1/b}^T\Gamma_{1/b\cdot}$$

As usual, only the first element of $\underset{\rightarrow}{\Delta u}$ (Δu_t) is implemented and the computation is repeated at the next sampling instant. Δu_t can be defined through eqn. (5.18).

It is clear, however, that there are implementation difficulties associated with the control law given in eqn. (5.23), in that the matrices involved in the formation of B_a, B_b are infinite dimensional, and so the control law, as presented, cannot be computed. Fortunately, these infinite dimensional matrices appear as quadratic products so that B_a, B_b are finite dimensional; here, we develop convenient and computationally efficient means of computing B_a, B_b.

It is well known that it is possible to compute the sum of squares of infinite but stable sequence through the use of appropriate Lyapunov equations. This idea was deployed in [30] to minimise a GPC cost where *only* the output horizon was infinite; the input horizon was taken to be finite. Here we removed this limitation by deriving necessary and sufficient conditions (rather than just sufficient conditions) for stable (and infinitely long) input *and* output prediction pairs while keeping the number of degrees of freedom finite. Earlier work identified the future control increments as the degrees of freedom and therefore precluded the (practical) use of infinite input horizons. In the following sections, we show that Lyapunov techniques can be deployed to compute and minimise the cost J_{CaSC} of eqn. (5.22) despite the use of infinite output and input horizons. We also show that this can be achieved without the use of Lyapunov equations

and indeed leads to an implementation which can be more efficient.

5.2.2 IHSPC using a Lyapunov equation

The key, in this approach, is to find a means of summing the squares of a stable sequence derived from a transfer function $g(z)=n(z)/d(z)$. This involves expressing the infinite sum in terms of the first n_n+1 coefficients only. Thus assume that $g(z)=\Sigma_{i=0}^{\infty} g_i z^{-i}$, then using $d(z)g(z)=n(z)$, the following recursive relationship holds for all g_i, $i>n_n$:

$$\begin{bmatrix} g_i \\ g_{i+1} \\ \vdots \\ g_{i+n_d-1} \end{bmatrix} = -C_d^{-1}\hat{H}_d \begin{bmatrix} g_{i-n_d} \\ g_{i-n_d+1} \\ \vdots \\ g_{i-1} \end{bmatrix} \tag{5.24}$$

where $\hat{H}_d$ is the hankel matrix, H_d with the order of its columns reversed. Hence, it can be seen that with $M=C_d^{-1}\hat{H}_d$,

$$\sum_{i=0}^{\infty} g_i^2 = \left[\sum_{i=0}^{n_n-n_d} g_i^2\right] + [g_{n_n-n_d+1}\ g_{n_n-n_d+2}\ \cdots\ g_{n_n}] S \begin{bmatrix} g_{n_n-n_d+1} \\ g_{n_n-n_d+2} \\ \vdots \\ g_{n_n} \end{bmatrix}; \qquad S=\sum_{i=0}^{\infty} (M^T)^i M^i \tag{5.25}$$

It is easy to show that S satisfies the Lyapunov equation, $S=I+M^TSM$ and hence the infinite sum of eqn. (5.25) can be easily evaluated. Thus, if the coefficients of $g(z)$ are given as the elements of the infinite length vector, $\boldsymbol{v}_g=\Gamma_{1/d}\boldsymbol{v}_n$ (where the elements of $\boldsymbol{v}_n$ are the coefficients of $n(z)$), we may write eqn. (5.25) in matrix form as:

$$\begin{aligned} \|\boldsymbol{v}_g\|_2^2 &= \boldsymbol{v}_g^{(1)T}\boldsymbol{v}_g^{(1)} + \boldsymbol{v}_g^{(2)T} S \boldsymbol{v}_g^{(2)} \\ &= \boldsymbol{v}_n^T \Gamma_{1/d}^{(1)T}\Gamma_{1/d}^{(1)}\boldsymbol{v}_n + \boldsymbol{v}_n^T \Gamma_{1/d}^{(2)T} S \Gamma_{1/d}^{(2)}\boldsymbol{v}_n \\ &= \boldsymbol{v}_n^T\left(\Gamma_{1/d}^{(1)T}\Gamma_{1/d}^{(1)} + \Gamma_{1/d}^{(2)T} S \Gamma_{1/d}^{(2)}\right)\boldsymbol{v}_n \\ &= \boldsymbol{v}_n^T \Gamma_{1/d}^T \Gamma_{1/d}\boldsymbol{v}_n = \boldsymbol{v}_n^T B_d \boldsymbol{v}_n \end{aligned} \tag{5.26}$$

where $\boldsymbol{v}_g^{(1)}$ contains the first n_n-n_d+1 elements of $\boldsymbol{v}_g$, and $\boldsymbol{v}_g^{(2)}$ contains the next n_d

elements, and thus $\Gamma_{1/d}^{(1)}$, $\Gamma_{1/d}^{(2)}$ contain the first n_n-n_d+1, next n_d rows of $\Gamma_{1/d}$; we note that if $n_d > n_n$, then $\Gamma_d^{(1)}$ would be empty and $\Gamma_d^{(2)}$ can simply be augmented on top by n_d-n_n-1 rows of zeros.

These results can be used to evaluate B_a, B_b of eqn. (5.23) simply by taking the polynomials $n(z)$, $d(z)$ to be the numerator and denominator polynomials of eqn. (5.10)a first and then eqn. (5.10)b second. Note that the order of the numerators, denominators of eqns. (5.10)a,b are $\max[n_{b^-}+n_c-1,\ n_r+n_{a^-}-2]$, n_{a^-}, and $n_{a^-}+n_c$, n_{b^-} respectively. So, defining $\Gamma_{1/a^-}^{(1)}$, $\Gamma_{1/a^-}^{(2)}$, $\Gamma_{1/b^-}^{(1)}$, $\Gamma_{1/b^-}^{(2)}$ as appropriate partitions of the first $\max[n_{b^-}+n_c,\ n_r+n_{a^-}-1]$, $n_c+n_{a^-}+1$ rows of the Toeplitz matrices Γ_{1/a^-}, Γ_{1/b^-} of eqns. (5.13)a,c and M_a, M_b from $M_a=C_{a^-}^{-1}\hat{H}_{a^-}$, $M_b=C_{b^-}^{-1}\hat{H}_{b^-}$ and solving for S_a, S_b from $S_a=I+M_a^T S_a M_a$, $S_b=I+M_b^T S_b M_b$, we may write B_a, B_b as:

$$B_a=\Gamma_{1/a^-}^{(1)T}\Gamma_{1/a^-}^{(1)}+\Gamma_{1/a^-}^{(2)T}S_a\Gamma_{1/a^-}^{(2)};\qquad B_b=\Gamma_{1/b^-}^{(1)T}\Gamma_{1/b^-}^{(1)}+\Gamma_{1/b^-}^{(2)T}S_b\Gamma_{1/b^-}^{(2)} \tag{5.27}$$

5.2.3 IHSPC without using a Lyapunov equation

The previous section shows that B_a, B_b can be calculated indirectly through the solution of a Lyapunov equation, however, it is easy to identify the coefficients of these quadratic matrices directly.

Lemma 5.1 Let $d(z)$ be a polynomial with all its roots inside the unit circle. Then the i,j element, $i \geq j$ (the argument for $i<j$ is identical) of the matrix $\Gamma_{1/d}^T\Gamma_{1/d}$, is given by the coefficient of z^{i-j} in the Laurent series expansion about $z=0$ of $f(z)=1/\mathrm{d}^*(z)\mathrm{d}(z)$, where the * superscript changes negative powers of z to positive (causal to anti-causal).

Proof: The j^{th} column of $\Gamma_{1/d}$ contains the coefficients of a Taylor series expansion in

terms of z^{-1} of $z^{j+1}/d(z)$. This is exactly equivalent to the coefficients of a Taylor series expansion in terms of z of $z^{j-1}/d^*(z)$. Let these expansions be:

$$\frac{1}{d(z)} = \sum_{i=0}^{\infty} g_i z^{-i}; \qquad \frac{1}{d^*(z)} = \sum_{i=0}^{\infty} g_i z^{i} \tag{5.28}$$

Hence $f(z) = (\Sigma_{i=0}{}^{\infty} g_i z^{-i})(\Sigma_{i=0}{}^{\infty} g_i z^{i})$, or:

$$f(z) = \sum_{i=-\infty}^{\infty} f_i z^{i}; \qquad \text{with} \qquad f_i = f_{-i} = \sum_{j=i}^{\infty} g_j g_{j-i} \tag{5.29}$$

The proof is completed by noting that the i^{th} row of $\Gamma_{1/d}{}^T$ is given by $[0,\ldots,0,g_0,g_1,\ldots]$ and the j^{th} column of $\Gamma_{1/d}$ is given by $[0,\ldots,0,g_0,g_1,\ldots]^T$, so that the i,j element of $\Gamma_{1/d}{}^T\Gamma_{1/d}$ is given by $g_0g_{i-j}+g_1g_{i-j+1}+\ldots$ which is identical to the definition of f_{i-j} given in eqn. (5.29). □

Remark 5.3 $f_i=f_{-i}$, so if $\Gamma_{1/d}$ has $m+1$ columns, then there are only $m+1$ distinct values of f_i in the matrix $\Gamma_{1/d}{}^T\Gamma_{1/d}$. Therefore, the computation of this matrix reduces to the computation of the first $m+1$ coefficients of the Laurent series expansion of $f(z)$.

We now propose a numerically efficient means of computing the first $m+1$ coefficients of the Laurent expansion of $f(z)$:

Equating the coefficients in z^i, $(0 \le i \le m)$ of $1/d(z)$ to those of $f(z)d^*(z)$ gives the following independent set of $m+1$ linear equations:

$$\left[\begin{array}{ccccc|c|cccc|c} d_0 & d_1 & \cdots & d_{n_d} & \cdots & 0 & 0 & \cdots & \cdots & 0 & \cdots \\ 0 & d_0 & \cdots & d_{n_d-1} & \cdots & 0 & 0 & \cdots & \cdots & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & d_0 & d_1 & \cdots & d_{n_d} & 0 & \cdots \end{array}\right] \left[\begin{array}{c} f_m \\ \vdots \\ f_1 \\ \hline f_0 \\ \hline f_1 \\ \vdots \\ f_m \end{array}\right] = [A_1|A_2|A_3] \left[\begin{array}{c} \hat{f} \\ f_0 \\ f \end{array}\right] = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ \frac{1}{d_0} \end{array}\right] \tag{5.30}$$

The coefficients of $f(z)$ can then be computed from:

$$[A_2|\hat{A}_1+A_3] \left[\begin{array}{c} f_0 \\ \hline f \end{array}\right] = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ \frac{1}{d_0} \end{array}\right] \tag{5.31}$$

where $\hat{A}_1$ is A_1 with the column order reversed; eqn. (5.31) is clearly trivial to solve. It is noted that the minimum number of coefficients that can be computed is n_d+1.

Theorem 5.2 The elements of the matrices B_a and B_b can be computed by setting $d(z)$ equal to $a^-(z)$ and $b^-(z)$ in the formulation of eqn. (5.31).

Proof: This is an obvious consequence of Lemma 5.1, the development which follows the lemma, and the definitions of B_a and B_b as given in eqns. (5.23)b,c.

This section and the previous provide two ways to compute B_a, B_b; a comparison of their relative merit follows.

The Lyapunov approach defines B_a, B_b from eqns. (5.27), requiring: i) computation of M_a, M_b (approx. n_a^3-, n_b^3- multiplications); ii) computation of S_a, S_b which involves the solution of $(n_a^2+n_a)/2$, $(n_b^2+n_b)/2$ linear equations; iii) computation of $\Gamma^{(1)}_{1/a^-}$, $\Gamma^{(2)}_{1/a^-}$, $\Gamma^{(1)}_{1/b^-}$, $\Gamma^{(2)}_{1/b^-}$ (approx. $n_a(n_b+n_c-1)$, $n_b(n_a+n_c)$ multiplications); and iv)

computation of B_1, B_2 (approx. $2n_{a^-}^3$, $2n_{b^-}^3$ multiplications).

The approach of Theorem 5.2, on the other hand, requires the solution of m_b+1, m_a+1 linear equations of the form (5.31) where $m_b=\max[n_c+n_{b^+}-1, n_r+n_{a^-}-2, n_{a^-}]$ and $d(z)=a^-(z)$, and $m_a=\max[n_c+n_{a^+}, n_{b^-}]$ and $d(z)=b^-(z)$.

It is clear that, depending on the various magnitudes of n_{a^-}, n_{b^-}, n_{a^+}, n_{b^+}, n_c, either method of computation can be more efficient. In particular, if n_c is small, but n_{a^-}, n_{b^-} are large, then one would favour the approach of Theorem 5.2, as the solution of fewer linear equations is required.

5.2.4 Simulation examples

In this section we demonstrate the efficacy of the IHSPC algorithm by comparing it with: (i) The original GPC algorithm as presented in [7], and (ii) RM, the algorithm of [30], each with the same number of degrees of freedom. The algorithms will be compared by way of simulation plots (the plots for IHSPC in solid line, RM in dashed line and GPC in dash-dotted lines) and the measure of performance:

$$J_{run}=\sum_{i=0}^{runtime} e_i^2+\lambda\Delta u_i^2 \qquad \textbf{(5.32)}$$

Example 5.3 Let the system model be defined by $a(z)=1-2.5z^{-1}+z^{-2}$ and $b(z)=1-0.7z^{-1}$, which has an unstable pole at $z=2$; and let the control parameters be $n_c=1$, and $\lambda=0.1$. The output/input plots are given in Figure 5.4 and the measures of runtime performance are given in Table 1; both illustrate the improvements gained by IHSPC.

GPC	RM	IHSPC
0.3092	0.3043	0.2875

Table 5.1: Runtime costs for Example 5.3

Example 5.4 For this example, let $a(z)=1-5.5z^{-1}+8.54z^{-2}-3.2z^{-3}+0.24z^{-4}$ and $b(z)=1+0.1z^{-1}-3.1z^{-2}+1.4z^{-3}$, which has unstable poles at $z=2,3$ and a non-minimum phase zero at $z=-2$; and let $n_c=1$, and $\lambda=1$. The output/input plots are given in Figure 5.5 and the runtime performance is presented in Table 2. Again, the improvement is clear, and in fact, GPC is unstable.

GPC	RM	IHSPC
∞	9.5993	7.8363

Table 5.2: Runtime costs for Example 5.4

It is noted that better results for GPC can be obtained for different combinations of control and output horizons. However, for this example (for $\lambda=0.1$): i) GPC cannot stabilize the model for a control horizon of one and two; and ii) for a control horizon of three, too small an output horizon gave instability and too large an output horizon gave numerical problems (as the model is open-loop unstable). Reasonable, though inferior, results could be obtained for a limited range of output horizons around 15. By contrast, both RM and IHSPC have guaranteed stability and, moreover, gave good performance for any control horizon.

Thus far, this chapter makes two main contributions. First, it presents an efficient means of classifying pairs of stable input/error predictions in a manner which makes transparent the remaining degrees of freedom, and second, it proposes a framework for the use of these predictions in an infinite horizon stable predictive control law. The conditions used to classify the predictions are both necessary and sufficient for stability, whereas earlier work used sufficient conditions only. In particular, here we allow the input trajectories to be infinite sequences, whereas other work in this area forces the input trajectory to have a finite (and usually small) number of changes. The use of necessary conditions has two obvious benefits: i) it releases as many degrees of freedom as possible for meeting performance criteria; and ii) it is more likely to be feasible when subject to system input constraints for small numbers of degrees of freedom. Clearly, this latter point can be used to significant computational advantage and/or makes a feasibility assumption easier to meet.

Finally, it was noted that most authors use a Lyapunov equation to calculate performance indices with ILS outputs, but FLS inputs. Here, we have extended the approach to performance indices with ILS outputs and inputs, and we propose an alternative approach which is particularly suitable for the types of problems arising in infinite horizon GPC and, in most cases, will be computationally more efficient.

5.3 Application summary - tokamak plasma control

The results of the previous sections have been applied to the problem of controlling the vertical position of plasma in the Compass-D tokamak. This application has been reported elsewhere ([47], [48]), and is briefly summarized in this thesis to provide a real

world example of the application of these results; all the other examples in this thesis are purpose built simulations to illustrate specific results.

Tokamaks are torus-shaped plasma containment devices which could form the basis for the nuclear fusion power plants of the future; they use magnetic fields to confine a plasma of ionized hydrogen atoms while it is heated to (hopefully) fusion temperatures. The Compass-D tokamak is a medium-sized test device which is used to study the instabilities and to establish the control techniques necessary for larger tokamak devices like the currently existing JET (Joint European Torus) and the future ITER (International Thermonuclear Experimental Reactor). ITER is in the design stage and plasma control is an area of great importance and some concern.

At desired operating points, the plasma vertical position is unstable and is currently controlled in the Compass-D tokamak with an analogue P+D control scheme. This scheme uses plasma velocity sensors located both inside and outside the vacuum vessel, but in larger devices, like ITER, the environment inside the vacuum vessel will be hostile to internal sensors. The P+D controller cannot stabilize the vertical position without these internal sensors, and thus a controller which uses only external sensors is required. Additionally, in all previous experiments, the vertical position control signals were dominated by a 600 Hz component due to interference from actuators used in other control loops, so another requirement of any new controller is that it reject this interference. The model of the plasma vertical position is obtained from experimental data which is inherently noisy and which reveals different open-loop growth rates at different operating points, and thus a further requirement is that the controller be robust to model uncertainty.

The plasma position response, as measured by the external sensors, was identified

from experimental data using a least squares fit and the SGPC model, $G(z)=z^{-1}b(z)/a(z)$ (including actuator and plant), for the desired operating point was found to be [47]:

$$a(z)=1-2.4519z^{-1}+1.5786z^{-2}+0.1848z^{-3}-0.3678z^{-4}+0.0832z^{-5}-0.0272z^{-6} \quad \textbf{(5.33)}$$
$$b(z)= \quad 10^{-3}\ (0.2239z^{-2}+0.5921z^{-3}+0.3646z^{-4}+0.0403z^{-5})$$

which has an unstable pole at $z=1.1099$, a slow pole at $z=0.9940$, and a non-minimum phase zero at $z=-1.7917$. Two controllers were designed; the first using standard H_∞ design methodology with the controller, $K(z)$, in the forward loop and negative unity feedback, and the second was IHSPC. The H_∞ controller was designed so that the modulus of the frequency response of the sensitivity transfer function, $S(z)=1/(1+G(z)K(z))$, was kept small at lower frequencies to ensure small position errors (performance), the modulus of the frequency response of $K(z)S(z)$ was kept small at 600 Hz to ensure rejection of the known interference (noise rejection), and the modulus of the frequency response of the closed-loop transfer function, $T(z)=G(z)K(z)S(z)$, was kept small at higher frequencies to ensure good stability robustness margins (robustness to multiplicative model uncertainty).

The IHSPC controller was designed in two stages. The first stage optimized performance using the results of Section 5.2 with the model modified to place a 600 Hz notch filter, $f(z)=f_n(z)/f_d(z)$, in the plant to reject the known interference; thus, the model was defined from $b'(z)=f_n(z)b(z)$ and $a'(z)=f_d(z)a(z)$, but of course the cost was chosen to penalize actual plant input increments. $a^+(z)$ and $b^+(z)$ were defined to include all roots on or outside a circle of radius $\rho=0.9$ and the other parameters were chosen to be $n_c=50$ and $\lambda=1000$. The closed-loop pole polynomial, $p_c(z)$, was established from eqn. (2.23)c with $D_k(z)$ and $N_k(z)$ determined from the explicit solution to the optimum IHSPC cost using results analogous to those of Section 5.1.3.

The second design stage optimized robustness and noise handling properties as per Section 5.1.3. The plasma vertical position of the Compass-D operates at a constant set-point with an adjustable reference; thus, zero steady-state error is not a design requirement. Therefore, IHSPC's explicit integrator was removed from the controller denominator polynomial; so that optimum performance was unaffected, the controller denominator and numerator polynomials, $D(z)$ and $N(z)$, were chosen to satisfy $p_c(z)=a(z)D(z)+z^{-1}b(z)N(z)$. With particular solutions, $D_p(z)$ and $N_p(z)$, the appropriate class of controller polynomials was defined as $D(z)=D_p(z)-z^{-1}b(z)Q(z)$ and $N(z)=N_p(z)+a(z)Q(z)$, where $Q(z)$ is an arbitrary polynomial. $Q(z)$ was chosen to optimize noise rejection and robustness to additive model uncertainty by using the procedures given in Section 2.2; large penalties were placed at 600 Hz to preserve the effect of the notch filter and at the approximate gain cross-over frequency to ensure a large stability margin.

By way of comparison of the resulting H_∞ and IHSPC controllers for the nominal plant, we provide the simulations of Figure 5.6-Figure 5.8. The modulus of the frequency response of $K(z)S(z)$ is plotted in Figure 5.6, and a Nyquist diagram of the loop gain, $L(z)=G(z)K(z)$, is shown in Figure 5.7; in both figures, the H_∞ controller is indicated by solid lines, the IHSPC controller (with $Q(z)=0$) is indicated with dashed lines, and the optimum IHSPC controller is indicated with dotted lines. The H_∞ and optimum IHSPC controllers obviously have very similar loop gain properties. The step responses of these two controllers are shown in Figure 5.8; we note, however, that good step response properties were not design objectives. The H_∞ responses are more oscillatory than those of IHSPC, have significantly larger overshoots, and the input response is significantly more active.

Because IHSPC has decoupled the design process, it achieves optimum

performance in a two-norm sense, and then, without affecting this optimality, it uses remaining degrees of freedom to optimize robustness and noise rejection properties. Thus, IHSPC provides obvious advantages in combining two- and infinity-norm specifications in such a way that both are optimal; we note, though, that the performance parameters (ρ, n_c, and λ) must be chosen carefully to ensure that the remaining degrees of freedom are adequate to meet other requirements like robustness and noise rejection. The H_∞ methodology seems to allows for a more direct trade-off between performance and robustness, but this, of course, still depends on judicious choice of weights, and the performance measure is the infinity-norm.

Both controllers were implemented on the Compass-D with a digital signal processor (DSP), and as the reference signal was constant, they produced similar results. Both controllers successfully stabilized the plasma vertical position using only external sensors and rejected the 600 Hz noise, meeting all design objectives. Further details of these results are reported in [47] and [48].

5.4 Constrained Cautious Stable Predictive Control

Thus far, we have concentrated on the cost portion of Cautious Stable Predictive Control; now we turn to the problem of enforcing input constraints over infinite input horizons.

5.4.1 Constraint checking with infinite input horizons

We return to the output/input (rather than error/input) form of ILS prediction equations (3.39) of Section 3.2.1, which as z-transforms, can be written as:

$$
\begin{aligned}
y(z) &= \frac{b^+(z)}{a^-(z)}c(z) + \frac{z^{-n_c}b^+(z)}{\Delta(z)a^-(z)}c_\infty + \frac{1}{a^-(z)}\phi_p(z) \\
\Delta u(z) &= \frac{\alpha^+(z)}{b^-(z)}c(z) + \frac{z^{-n_c}a^+(z)}{b^-(z)}c_\infty + \frac{1}{b^-(z)}\psi_p(z) \\
u(z) &= \frac{a^+(z)}{b^-(z)}c(z) + \frac{z^{-n_c}a^+(z)}{\Delta(z)b^-(z)}c_\infty + \frac{1}{\Delta(z)b^-(z)}\psi_p(z) + \frac{1}{\Delta(z)}u_{t-1}
\end{aligned}
\qquad \textbf{(5.34)}
$$

5.4.1.1 Definition of problem

Most practical control systems are subject to (hard) input constraints such as:

$$
-R = \underline{\Delta u} \le \Delta u_{t+i} \le \overline{\Delta u} = R; \qquad U_o - U = \underline{u} \le u_{t+i} \le \overline{u} = U_o + U \qquad i \ge 0 \qquad \textbf{(5.35)}
$$

where it is assumed (as is reasonable) that the steady-state input/control increment required to maintain the set-point are on the interior of the constraint intervals, namely that they are at least a distance ϵ (for ϵ arbitrarily small) inside the above limits:

$$
\underline{u} + \epsilon \le u_{ss} = \frac{a(1)}{b(1)}r \le \overline{u} - \epsilon; \qquad \underline{\Delta u} + \epsilon \le 0 \le \overline{\Delta u} - \epsilon \qquad \textbf{(5.36)}
$$

and for convenience (only) we have assumed that the input limits are time invariant. The input constraints above may come into conflict with the use of terminal constraints, and we have termed this condition short term infeasibility (STIF). By contrast, the compatibility of terminal constraints with input constraints is referred to as short term feasibility (STF).

CaSC does not take constraints (5.35) into consideration, and hence the optimal predicted values for u and/or Δu may lie outside the limits of (5.35); this will lead to sub-optimality and may even result in instability. Thus, it is important to incorporate constraints (5.35) into the optimization problem; earlier work (eg [26], [46], [33]) achieves this through the use of quadratic programming. Work to date has restricted attention to finite length sequences for the future Δu's; this is a consequence of the fact

that terminal constraint (5.12)b has been invoked on Δu rather than on $\Delta \hat{u}$. Terminal constraints of this sort are convenient in establishing stability, which depends on a guarantee of feasibility, ie a guarantee that the terminal constraints can be met within the input constraints. However, such terminal constraints are only sufficient for the stability of predicted trajectories and thus may result in an unnecessary restriction of the degrees of design freedom. This could have a significant effect on performance; furthermore, on account of the feasibility requirement it may necessitate the use of longer control horizons with a concomitant significant increase in the quadratic programming computational load.

Here our concern is use the maximum degrees of design freedom possible and this implies the need for conditions such as those developed in the previous section which are both necessary and sufficient. The difficulty with these, however, is that they result in future control trajectories which form infinite length sequences; at first sight this requires that constraint satisfaction be tested over an infinite horizon. To overcome this, problem here we develop some efficient (albeit loose) bounding results which enable the definition of a finite horizon, n_{con}, referred to as the constraints horizon, which has the property that constraint satisfaction over n_{con} implies actual constraint satisfaction over an infinite horizon. It is noted that the reconciliation of finite with infinite constraint horizons has been addressed elsewhere ([30], [54]), but in a different context (related to output constraints) using different (state-space) bounding techniques. The techniques developed here address a new problem, namely the maximization of available control freedom with the view to satisfying the feasibility requirement as well as improving performance and/or reducing the computational burden.

5.4.1.2 Bounding conditions

We wish to invoke terminal constraints (5.12) without violating physical limits (5.35), and as remarked above the difficulty here is that future control trajectories are ILS's. So now we seek to determine a finite (and preferably small) value for n_{con} such that constraint satisfaction for all future times is guaranteed by constraint satisfaction up to n_{con}. An obvious way to achieve this is to: (i) consider a particular future time instant $t+i$; (ii) derive bounds on the maximum and minimum values of the predicted u's and Δu's beyond this time instant $t+i$, namely bounds for future values at all times $t+j, j \geq i$; and (iii) increase i until the bounds of (ii) are within the physical limits of (5.35).

Remark 5.4 The bounds used in the determination of n_{con} need not be "tight", in that "loose" bounds would merely result in a conservative choice for n_{con}. The result of this is that (5.35) will be checked at more time instants than necessary. This will not have a significant effect on computations because the extra checks will correspond to constraint inequalities which, by definition, will be inactive: if future u's and Δu's are within "loose" bounds, which themselves are within the limits of (5.35), then clearly such u's and Δu's will satisfy (5.35) *a fortiori*. For this reason, the emphasis in what follows will be on ease of presentation/computation rather than obtaining the tightest bounds possible. Along the same lines, we shall refrain from defining different values for n_{con} for each of conditions (5.35)a,b.

From prediction eqns. (3.39)b,c, it is apparent that the future predicted values of u and Δu depend on future values of c which, as yet, are unknown. In determining bounds on u and Δu, therefore, we must first stipulate bounds on c.

Lemma 5.2 Input constraints (5.35) will be violated if any of the future c's lie outside the respective interval defined below:

$$\underline{c}_i \le c_i \le \bar{c}_i \qquad i=0,\ldots,n_c-1$$

$$\bar{c}_i=\min\left[\sum_{j=1}^{n_c}(\max\ [S_{ij}\overline{\Delta u},\ S_{ij}\underline{\Delta u}]-S_{ij}s_j),\ \sum_{j=1}^{n_c}(\max\ [V_{ij}\bar{u},\ V_{ij}\underline{u}]-V_{ij}v_j)\right] \qquad \textbf{(5.37)}$$

$$\underline{c}_i=\max\left[\sum_{j=1}^{n_c}(\min\ [S_{ij}\overline{\Delta u},\ S_{ij}\underline{\Delta u}]-S_{ij}s_j),\ \sum_{j=1}^{n_c}(\min\ [V_{ij}\bar{u},\ V_{ij}\underline{u}]-V_{ij}v_j)\right]$$

where S_{ij} and V_{ij} denote the ij^{th} elements of matrices S and V, whereas s_j, v_j denote the j^{th} element of vectors s, v; S and V are defined as the inverses of block matrices formed out of the first n_c rows of Γ_{α^+/b^-}, and Γ_{a^+/b^-}, respectively and s, v are the vectors formed out of the first n_c elements of $\Gamma_{1/b^-}P_2\boldsymbol{p}$, and $\Gamma_{1/\Delta b^-}P_2\boldsymbol{p}+u_{t-1}\mathbf{1}$.

Proof: With the definitions of S, V, s and v, the first n_c scalar equations implied by eqn. (3.39)b,c can be rearranged to give

$$\underset{\rightarrow}{c}=V\left(\begin{bmatrix}u_t\\ \vdots\\ u_{t+n_c-1}\end{bmatrix}-\boldsymbol{v}\right);\qquad \underset{\rightarrow}{c}=S\left(\begin{bmatrix}\Delta u_t\\ \vdots\\ \Delta u_{t+n_c-1}\end{bmatrix}-\boldsymbol{s}\right) \qquad \textbf{(5.38)}$$

where use has been made of the fact that, due to the presence of z^{-n_c} in the second term of eqns. (5.34)b,c, the first n_c elements of $\theta_{\alpha^+/b^-}c_\infty$, $\theta_{a^+/b^-}c_\infty$ are zero. Thus, c_i is given as (i) the sum (over j) of terms $S_{ij}\Delta u_{t+j}-S_{ij}s_j$, and (ii) the sum (over j) of terms $V_{ij}u_{t+j}-V_{ij}v_j$. The proof is completed by invoking the limits of (5.35) on Δu_{t+j} and u_{t+j}; (i) and (ii) imply different intervals for c_i, and since both must hold true, the intersection of the two is used to define the bounds of (5.37). □

In order to derive bounds on the predicted Δu_{t+j} for $j \ge i$, consider the first term of the RHS of (5.34)b which is of the form $h(z)f(z)$, with $h(z)=\alpha^+(z)/b^-(z)$ and $f(z)=c(z)$. The

other two terms have the same form and the overall result can be assembled by a process of linear superposition; the only difference is that $f(z)$ for the second and third term are known, whereas $f(z)$ for the first term is unknown, but bounded. The lemmata below deal with these two different cases separately.

Lemma 5.3 Let $g(z)=h(z)f(z)$ where $h(z)$ is an asymptotically stable transfer function with impulse response $\{h_0,h_1,\ldots\}$ and $f(z)$ is a polynomial in z^{-1} of degree n_f whose coefficients f_i are known. Then bounds on each (and all subsequent) element of the impulse response of $g(z)$ are given by

$$\underline{G}_i \le g_j \le \overline{G}_i \quad (j \ge i) \qquad \begin{cases} \underline{G}_i = \sum_{k=0}^{n_f} \min\left[\underline{H}_{i-k} f_k,\ \overline{H}_{i-k} f_k\right] \\ \overline{G}_i = \sum_{k=0}^{n_f} \max\left[\underline{H}_{i-k} f_k,\ \overline{H}_{i-k} f_k\right] \end{cases} \qquad \begin{cases} \underline{H}_i = \min_l h_l \quad (l \ge i) \\ \overline{H}_i = \max_l h_l \quad (l \ge i) \end{cases} \tag{5.39}$$

Proof: This follows from the definition of $g(z)$ according to which g_i is given as the appropriate sum of products $h_{i-k}f_k$. □

Lemma 5.4 Let $g(z)=h(z)f(z)$ where $h(z)$ is an asymptotically stable transfer function with impulse response $\{h_0,h_1,\ldots\}$, and $f(z)$ is a polynomial in z^{-1} of degree n_f whose coefficients f_i are unknown, but bounded by $\underline{f}_i \le f_i \le \overline{f}_i$, and let the bounds, $\underline{H}_i$, and $\overline{H}_i$ be as defined in eqn. (5.39). Then bounds on each (and all subsequent) element of the impulse response of $g(z)$ are given by:

$$\underline{G}_i \le g_j \le \overline{G}_i \quad (j \ge i) \qquad \begin{cases} \underline{G}_i = \sum_{k=0}^{n_f} \min\left[\underline{H}_{i-k}\underline{f}_k,\ \overline{H}_{i-k}\underline{f}_k,\ \underline{H}_{i-k}\overline{f}_k,\ \overline{H}_{i-k}\overline{f}_k\right] \\ \overline{G}_i = \sum_{k=0}^{n_f} \max\left[\underline{H}_{i-k}\underline{f}_k,\ \overline{H}_{i-k}\underline{f}_k,\ \underline{H}_{i-k}\overline{f}_k,\ \overline{H}_{i-k}\overline{f}_k\right] \end{cases} \tag{5.40}$$

Proof: This is the same as for Lemma 5.3, except that, when dealing with the products

$h_{i-k}f_k$, we now have to consider intervals for both the coefficients of $h(z)$ and $f(z)$. □

Remark 5.5 The bounding results above afford a significant online computational advantage because bounds, $\underline{H}_i$, and $\overline{H}_i$ are time-invariant and thus can be calculated off line and saved in a look-up table. Therefore the bounds which apply over an infinite horizon are calculated by summing a finite (and indeed small) number of maxima/minima over two or four scalars.

As mentioned previously, the predicted Δu's and u's of eqn. (5.34)b,c can be viewed as the sum of products, $h(z)f(z)$, all of which can be bounded (as per Lemmata 5.3, 5.4) and then combined by linear superposition. To that end, we now define bounds for each part of eqns. (5.34)b,c, except for the last part of eqn. (5.34)c whose upper and lower bounds are obviously equal to u_{t-1}:

$h(z)$:	$\frac{\alpha^+(z)}{b^-(z)}$	$\frac{z^{-n_c}a^+(z)}{b^-(z)}$	$\frac{1}{b^-(z)}$	$\frac{a^+(z)}{b^-(z)}$	$\frac{z^{-n_c}a^+(z)}{\Delta(z)b^-(z)}$	$\frac{1}{\Delta(z)b^-(z)}$
$f(z)$:	$c(z)$	c_∞	$\psi_p(z)$	$c(z)$	c_∞	$\psi_p(z)$
Lemma:	5.4	5.3	5.3	5.4	5.3	5.3
Bounds:	$\underline{G}_i^1, \overline{G}_i^1$	$\underline{G}_i^2, \overline{G}_i^2$	$\underline{G}_i^3, \overline{G}_i^3$	$\underline{G}_i^4, \overline{G}_i^4$	$\underline{G}_i^5, \overline{G}_i^5$	$\underline{G}_i^6, \overline{G}_i^6$

Table 5.3 Definitions of bounds

It is then a simple matter to give bounds on the future elements of Δu and u as:

$$\underline{G}_i^1+\underline{G}_i^2+\underline{G}_i^3 \le \Delta u_{t+j} \le \overline{G}_i^1+\overline{G}_i^2+\overline{G}_i^3 \quad \underline{G}_i^4+\underline{G}_i^5+\underline{G}_i^6+u_{t-1} \le u_{t+j} \le \overline{G}_i^4+\overline{G}_i^5+\overline{G}_i^6+u_{t-1} \quad (j \ge i) \qquad \textbf{(5.41)}$$

Theorem 5.3 Let n_{con} be the smallest value of i for which the derived bounds are

"inside" the limits of input constraints (5.35), ie:

$$\begin{aligned} \underline{G}^1_{n_{con}} + \underline{G}^2_{n_{con}} + \underline{G}^3_{n_{con}} \geq \underline{\Delta u} \qquad & \underline{G}^4_{n_{con}} + \underline{G}^5_{n_{con}} + \underline{G}^6_{n_{con}} + u_{t-1} \geq \underline{u} \\ \overline{G}^1_{n_{con}} + \overline{G}^2_{n_{con}} + \overline{G}^3_{n_{con}} \leq \overline{\Delta u} \qquad & \overline{G}^4_{n_{con}} + \overline{G}^5_{n_{con}} + \overline{G}^6_{n_{con}} + u_{t-1} \leq \overline{u} \end{aligned} \tag{5.42}$$

Then if prediction eqns. (5.34)b,c are made to satisfy constraints (5.35) at $t+i$, for $i \leq n_{con}$, they will satisfy these constraints for all future times.

Proof: This follows from Lemmata 5.2-4 and the bound definitions of Table 5.3. □

Corollary 5.1 A finite n_{con} can always be found which satisfies Theorem 5.3.

Proof: Consider the bounds $\underline{G}^k_i$, $\overline{G}^k_i$ for k=1-3 associated with Δu_j, and apply the final value theorem to the corresponding $h(z)$ of Table 5.3 to deduce that, as $i \to \infty$, the elements h_i of the impulse response of $h(z)$ go to zero. As a consequence, the bounds $\underline{H}_i$, $\overline{H}_i$ (defined in eqn. (5.39)) will also converge to zero, and hence, so will the bounds on Δu_j of (5.41)a because of their definition as given in eqns. (5.39) or (5.40). Now, from (5.36)b, we know that $\underline{\Delta u}$, $\overline{\Delta u}$ are at least a distance ϵ away from 0; hence, for any ϵ (no matter how small), n_{con} can be chosen large enough so that the moduli of the bounds on Δu_j of conditions (5.41)a are less than ϵ: therefore conditions (5.42)a,c will hold true.

To complete the proof we need to show that (5.42)b,d hold true, and this can be established by similar arguments, except that now, by (5.36)a, we need to show that the bounds on u of (5.41)b converge to u_{ss}. This follows from:

(i) application of the final value theorem to (5.34)c according to which

$$u_\infty = \lim_{z \to 1} (1 - z^{-1}) u(z) = 0 + \frac{a^+(1)}{b^-(1)} c_\infty + \frac{\psi_p(1)}{b^-(1)} + u_{t-1} \tag{5.43}$$

(ii) application of the final value theorem to the bounds on u of (5.41)b which are the sum of the products $h(z)f(z)$ of the last three columns of Table 5.3 and u_{t-1}, according to

which:

$$\lim_{n_{con}\to\infty}\left(\underline{G}^4_{n_{con}}+\underline{G}^4_{n_{con}}+\underline{G}^4_{n_{con}}\right)+u_{t-1}=\lim_{n_{con}\to\infty}\left(\overline{G}^4_{n_{con}}+\overline{G}^4_{n_{con}}+\overline{G}^4_{n_{con}}\right)+u_{t-1}$$

$$=\lim_{z\to 1}(1-z^{-1})\left[\frac{\alpha^+(z)}{b^-(z)}c(z)+\frac{z^{-n_c}a^+(z)}{\Delta(z)b^-(z)}c_\infty+\frac{\psi_p(z)}{\Delta(z)b^-(z)}\right]+u_{t-1} \quad (5.44)$$

$$=0+\frac{a^+(1)}{b^-(1)}c_\infty+\frac{\psi_p(1)}{b^-(1)}+u_{t-1}$$

□

Algorithm 5.1 (Computation of n_{con})

Step 1: Set $i=n_u$.

Step 2: Compute $\underline{G}^k_i$, $\overline{G}^k_i$, k=1-6.

Step 3: Check the conditions of Theorem 5.3. If true, set $n_{con}=i$ and stop; otherwise, set $i=i+1$ and goto step 2.

5.4.2 The constrained CaSC algorithm

Here we present the Constrained Cautious Stable Control (CCaSC) algorithm and prove its stability property. For ease of presentation we consider first the simpler case where set-point changes are assumed not to lead to infeasibility. We then deal with the general case where the feasibility assumption is removed by a suitable change of terminal constraints; this change is identical to the modification of MCSGPC in that y is allowed to converge to some slack variable, s_∞, and s_∞ is then made subject to a slack variable end-point constraint.

5.4.2.1 The case of feasible set-point changes

Earlier work ([33], [53], [23], [30], [54]) has combined sufficient (but not

necessary) terminal constraints with input constraints to give stable predictive control algorithms. Here we do the same with respect to terminal constraints which are both sufficient and necessary.

Algorithm 5.2

Step 1: Invoke Algorithm 5.1 to compute n_{con}.

Step 2: Minimize over $\underset{\rightarrow}{c}$ the cost $\tilde{J}_{CaSC}$ of eqn. (5.11)c subject to input constraints (5.35) for $i \leq n_{con}$.

Step 3: Implement the first element of the corresponding vector of future u's given in eqn. (3.39)c; increment t and goto step 1.

Theorem 5.4 Assume that, at startup and at times of set-point changes, terminal and input constraints are consistent. Then the CCaSC algorithm is stabilizing and gives asymptotic tracking.

Proof: By the assumption, we have STF at startup, and Theorem 5.3 and Corollary 5.1 ensure that STF is also guaranteed at the next instant, because at least one feasible future control trajectory exists, namely that employed at the previous time instant. This argument can be propagated until t_s, the time of the next set-point change, but the assumption ensures STF at t_s, and thus the argument above can be extended for all times.

The rest of the proof relies on showing that the cost function $\tilde{J}_{CaSC}$ is a stable Lyapunov function. This is so because, at any time t, the terminal constraints ensure that the predictions for both $\tilde{e}_{t+n_y+i}$, $\Delta\hat{u}_{t+n_u+i}$ are zero for $i>0$; thus, the optimal $\tilde{J}_{CaSC}$ at t gives an upper bound on the value of $\tilde{J}_{CaSC}$ at $t+1$, and the cost cannot stay at this upper bound for more than a finite number of steps (namely the maximum of n_y and n_u), unless of

course the cost is zero. Thus, $\tilde{e}\to 0$, and $\Delta\hat{u}\to 0$; since $e(z)=\tilde{e}(z)/a^{-}(z)$ and $\Delta u(z)=\Delta\hat{u}(z)/b^{-}(z)$, where both $a^{-}(z)$ and $b^{-}(z)$ are stable, the same asymptotic property will apply with respect to e and Δu. □

5.4.2.2 The general case

The CaSC terminal output constraint, (5.12)a, can be rewritten as:

$$\tilde{y}_{t+i}=\tilde{r}_{t+n_r}=a^{-}(1)r_{t+t+n_r} \qquad \forall\ i\geq n_y \tag{5.45}$$

Viewed this way, $\tilde{y}$ must reach, within n_y steps, its demanded target $\tilde{r}_{t+n_r}=a^{-}(1)r_{t+n_r}$. On account of this implicit requirement, large set-point changes may necessitate the use of control moves which exceed the limits defined by the input constraints and thus may lead to infeasibility (STIF). As with MCSGPC, an obvious way to avoid this difficulty is to allow y to settle at s_∞, a degree of freedom, but to constrain s_∞ to converge to r_{t+n_r}. We implement this, as before, with the slack variable end-point constraint:

$$|s_\infty - s_\infty^{o}| \leq w|s_\infty^{old}-s_\infty^{o}| \qquad 0\leq w<1 \tag{5.46}$$

where s_∞^{old} is the optimal value of s_∞ computed at the previous time instant, and s_∞^{o} is the value of s_∞ closest to r_{t+n_r} which does not violate the input constraints; as seen later, the computation of s_∞^{o} simply involves a linear program.

This modification can be easily introduced into Algorithm 5.1 by including $c_\infty=a^{-}(1)s_\infty/b^{+}(1)$ of prediction eqns. (5.34) as an extra degree of freedom along with those contained in $c(z)$. However, the calculation of n_{con} has to be modified accordingly to accommodate the fact that c_∞ is unknown. This is easy to do providing that c_∞ is bounded, say by $\underline{c}_\infty$, $\bar{c}_\infty$, because then Algorithm 5.1 can be invoked exactly as given in Section 5.4.1.2, but with the bounds of $\underline{G}_i^k$, $\overline{G}_i^k$ for $k=2$ and $k=4$ computed as

dictated by Lemma 5.4 rather than Lemma 5.3. By (5.46), s_∞ must lie in the interval defined by s_∞^{old} and s_∞^{o} and therefore will also lie (*a fortiori*) in the interval defined by s_∞^{old} and r. The latter is more conservative than the former, but is preferred because, strictly speaking, the computation of s_∞^{o} requires prior knowledge of n_{con}. This simplification may result in larger than necessary values for n_{con}, but as per Remark 5.4, this is a matter of no computational significance. Thus, the bounds on c_∞ to be used in Lemma 5.4 will be taken to be:

$$\underline{c}_\infty = \min\left[\frac{a^-(1)}{b^+(1)}s_\infty^{old},\ \frac{a^-(1)}{b^+(1)}r\right] \qquad \bar{c}_\infty = \max\left[\frac{a^-(1)}{b^+(1)}s_\infty^{old},\ \frac{a^-(1)}{b^+(1)}r\right] \tag{5.47}$$

Remark 5.6 At first sight, it may appear that an unknown c_∞ would affect Lemma 5.2 (and hence, Lemma 5.4 also) since the vectors $\boldsymbol{v}$ and $\boldsymbol{s}$, of eqn. (5.38) themselves would be unknown. This is not true, because as pointed out in the proof of Lemma 5.2, c_∞ makes no contribution to the first n_c elements of the prediction vectors in (5.34). Equally, the use of a value for c_∞ which is not equal $a^-(1)r/b^+(1)$ does not affect adversely the proof of Corollary 5.1: conditions (5.42)a,c are unaffected, because the steady-state value of the bounds on Δu of condition (5.41)a do not depend on c_∞, whereas, following the procedure given in the proof of the corollary and by (5.47), it is easy to show that the steady-state value of the bounds on u of condition (5.41)b will be:

$$\frac{a(1)}{b(1)}s_\infty^{old}+\frac{\psi_p(1)}{b^-(1)}+u_{t-1} \quad \text{and} \quad \frac{a(1)}{b(1)}r+\frac{\psi_p(1)}{b^-(1)}+u_{t-1} \tag{5.48}$$

Clearly, both these values are within the physical limits of (5.35)a by an amount at least as large as the ϵ of eqn. (5.36).

Algorithm 5.3 (CCaSC)

Step 1: Compute n_{con} as per Algorithm 5.1, except that the bounds $\underline{G}_i^k$, $\overline{G}_i^k$ for k=2,4 now should be evaluated using Lemma 5.4 (invoking the limits on c_∞ of eqn. (5.47)) instead of Lemma 5.3.

Step 2: $\min_{\underset{\rightarrow}{c}, c_\infty} |r - s_\infty|$ subject to constraints (5.35) for $i \leq n_{con}$, and let the minimizing solution to this linear program for s_∞ be $s_\infty{}^o$.

Step 3: Minimize over $\underset{\rightarrow}{c}$ and c_∞ the cost $\tilde{J}_{CaSC}$ of eqn. (5.11)c subject to input constraints (5.35) for $i \leq n_{con}$ and to constraint (5.46).

Step 4: Implement the first element of the corresponding vector of future u's given in eqn. (3.39)c; increment t and goto step 1.

Theorem 5.5 Given feasibility at startup, CCaSC has guaranteed stability and asymptotic tracking.

Proof: Assume feasibility at a time t-1, so that, at t, $s_\infty{}^{old}$ defines a feasible choice for s_∞. On the other hand, $s_\infty{}^o$ is also feasible, since it is defined to be the value of s_∞ which is as close to r as is possible without violating the input constraints. Therefore, constraint (5.46) can be met and so replacing equality terminal constraint (5.12)a with inequality terminal constraint (5.46) guarantees that feasibility at t-1 implies feasibility at t for *any set-point change*. This argument propagates all the way back to startup which, by the assumption of the theorem, respects feasibility; thus, feasibility is guaranteed for all time.

The consequence of inequality constraint (5.46) is that, if $s_\infty{}^o \neq s_\infty{}^{old}$ for all future times, then s_∞ will converge asymptotically to r; therefore, by Theorem 5.4, we have stability and asymptotic tracking. This arguments holds even if $s_\infty{}^o = s_\infty{}^{old}$, so long as this

is not true for *all* future times; a case that can only arise if s_∞, s_∞^{old} and s_∞^{o} all "stagnate" at some value other than r. This of course cannot happen, because after n_{con} steps, the input constraints would hold with strict inequality, and so the linear program of step 2 of the algorithm would use the available control authority to move s_∞^{o} closer to r. □

5.4.3 Illustrative examples and comparisons

Earlier constrained predictive algorithms with guaranteed stability are based on conditions which are only sufficient for the stability of the prediction equations. The algorithms presented in this chapter are based on conditions which are both necessary and sufficient and therefore release as much control authority as is possible. The extra control authority thus generated can be deployed (i) in the further reduction of the cost J and hence improving performance, and/or (ii) in the reduction of the command horizon n_c (without violating feasibility) with a significant concomitant reduction in the computational burden involved in the application of constrained optimization routines such as quadratic programming. In this section, we illustrate these points by means of a numerical example which, by way of introduction to Chapter 6, shows how CCaSC compares with CSGPC, RM, and GPC in the presence of input constraints and disturbances.

Example 5.5 For this example, we use the model and control parameters of Example 5.1, but now assume that the inputs are subject to input constraints (5.35) with R=0.45, U_o=0, and U=1.5. We further assume that at t=3, a persistent (step) disturbance equal to 1/15 is encountered at the output of the plant. Figure 5.9 shows the output, input, and input increment responses for the four algorithms. CSGPC results are indicated with dotted lines, RM with dashed lines, GPC with dash-dotted lines, and CCaSC with solid

lines. CCaSC rejects the disturbance successfully, whereas the other algorithms all become unstable.

Thus, we note that CCaSC, as it uses necessary and sufficient conditions for stable predictions, is better able to cope with input constraints than either CSGPC or RM, which use sufficient only conditions. Furthermore, it also improves upon GPC, because GPC does not have a stability guarantee either in the unconstrained or constrained case.

5.5 Chapter summary

Terminal constraints which allowed the predicted output error to be an infinite length sequence have been explored before, but infinite length sequences for the control increments have not previously been considered. Clearly, this is impractical in the case where control increments are defined to be the degrees of freedom, and poses difficulties with respect to the enforcement of input constraints over an infinite horizon. In this chapter, we have overcome both of these problems by an appropriate characterization of the conditions which are both necessary and sufficient for the stability of predictions and by the introduction of appropriate bounds (which are easy to compute) on the predicted input and control increments. We have thus reserved the maximum possible control authority for the purpose of reducing computational complexity (through the use of smaller command horizons) and/or improving performance.

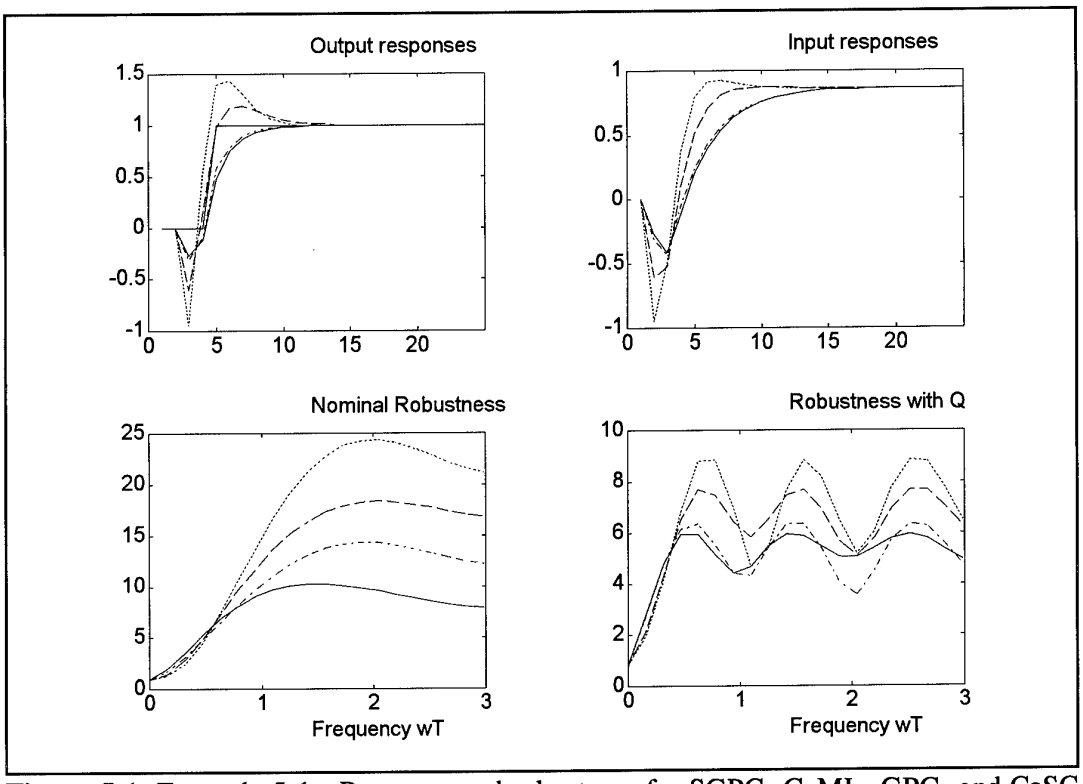


Figure 5.1 Example 5.1 - Response and robustness for SGPC, CaML, GPC, and CaSC

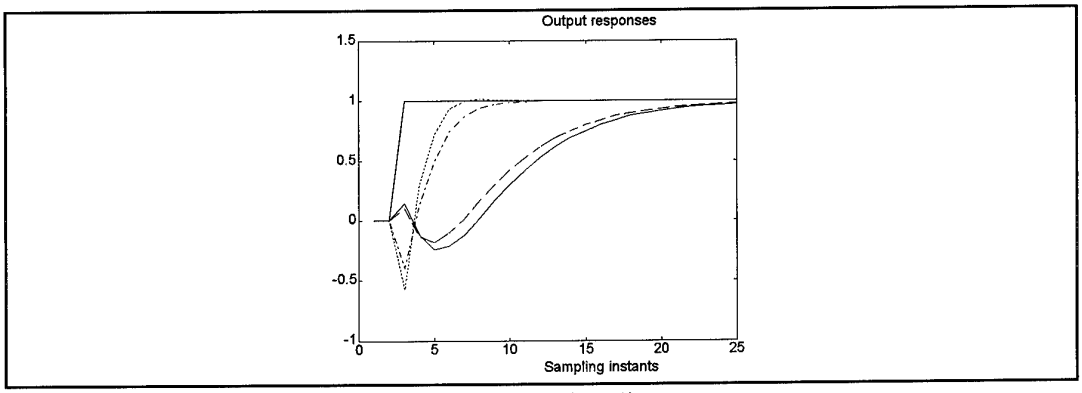


Figure 5.2 Example 5.2 - Output responses ($\rho=1$)

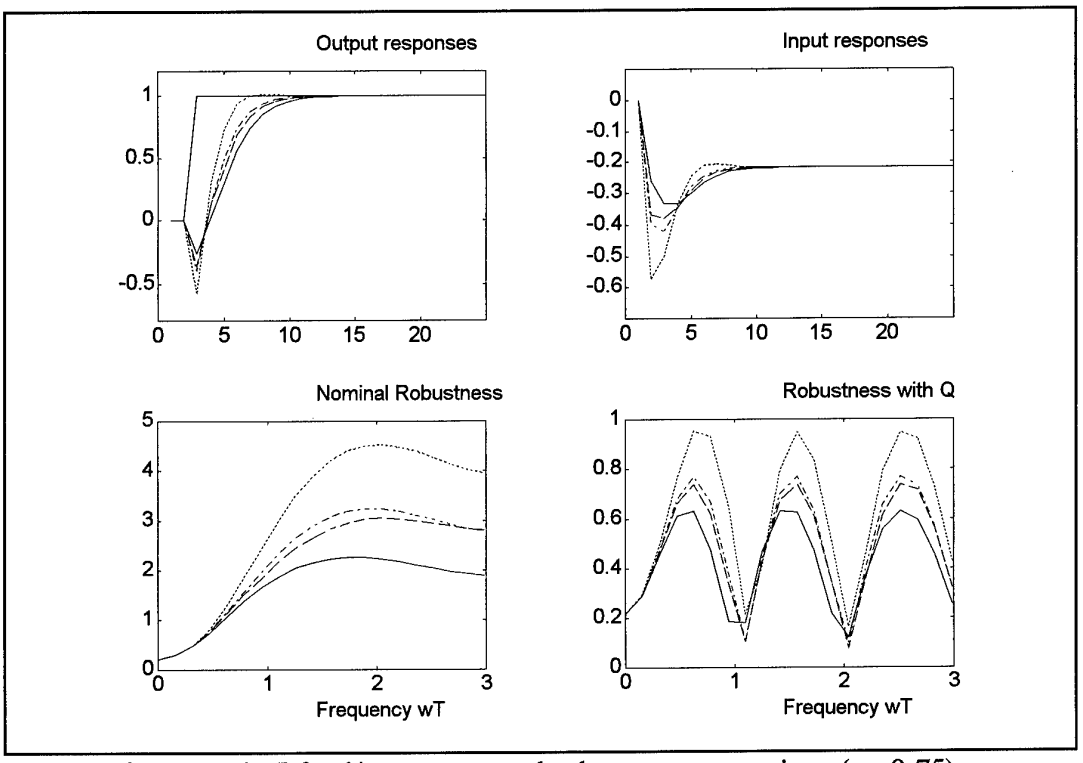


Figure 5.3 Example 5.2 - i/o response and robustness comparison (ρ=0.75)

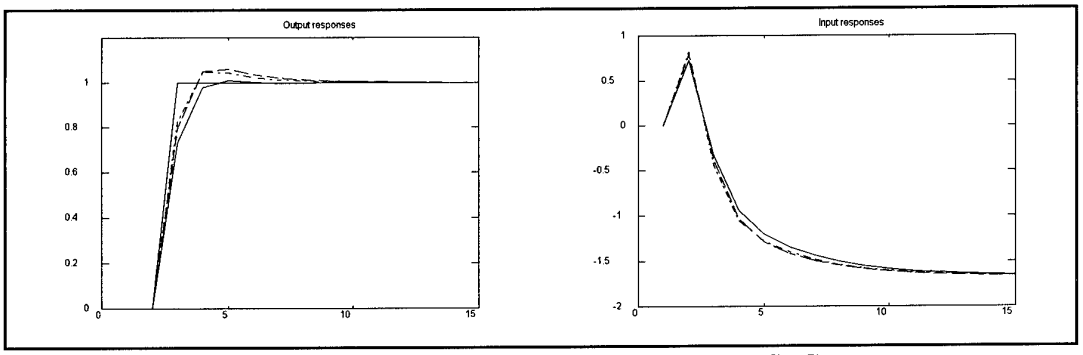


Figure 5.4 Example 5.3 - Comparison of GPC, RM, and IHSPC

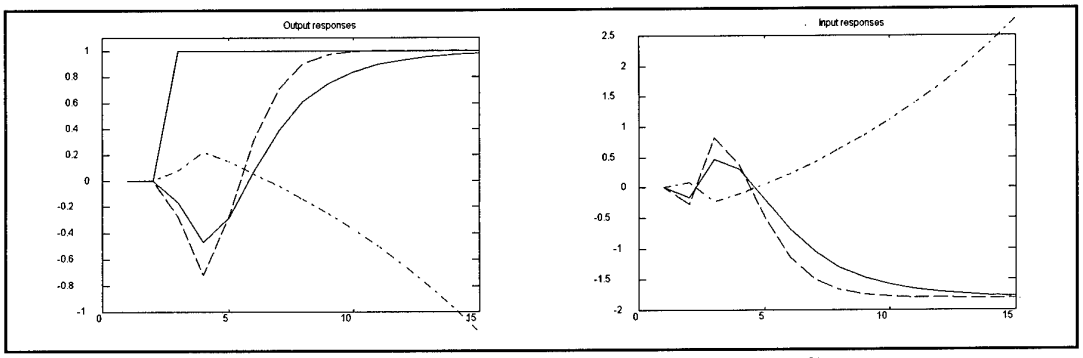


Figure 5.5 Example 5.4 - Comparison of GPC, RM, and IHSPC

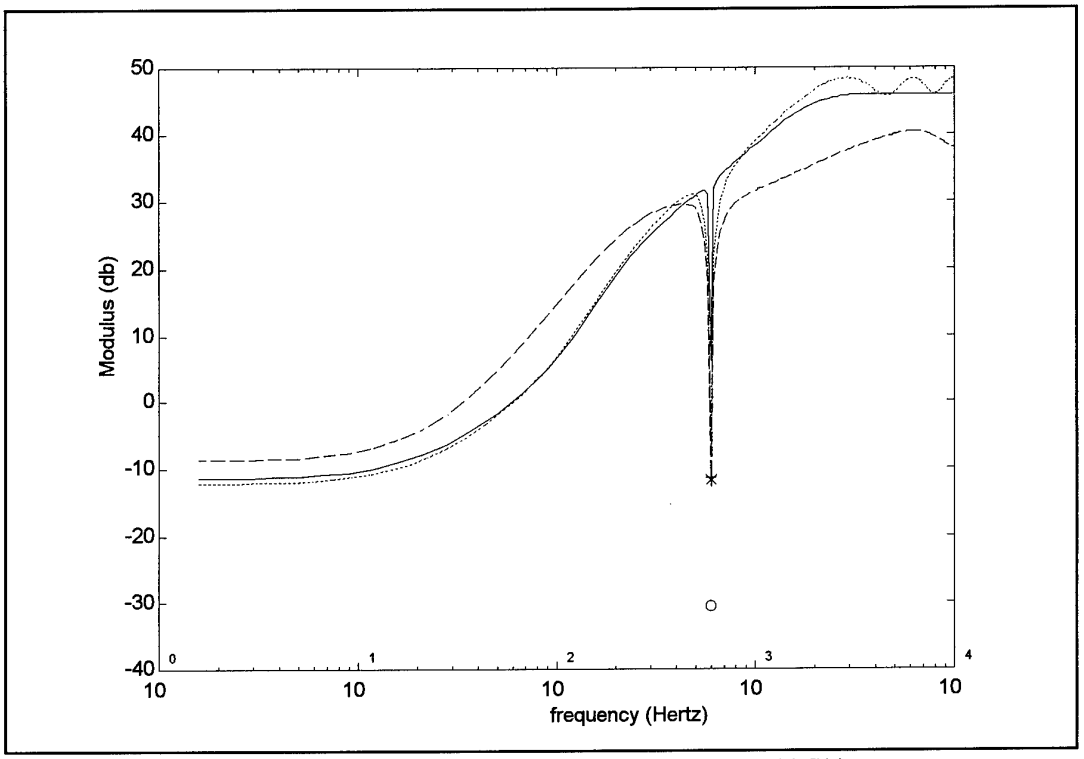


Figure 5.6 Tokamak Application - Modulus comparison of $K(z)S(z)$

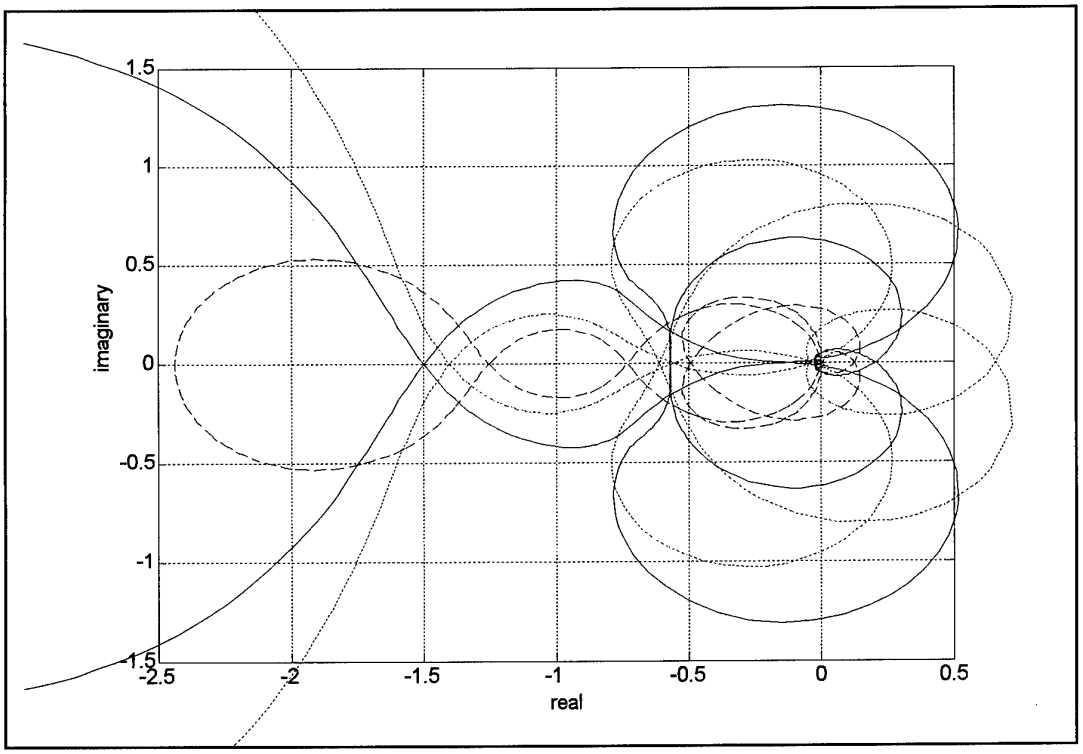


Figure 5.7 Tokamak Application - Nyquist comparison

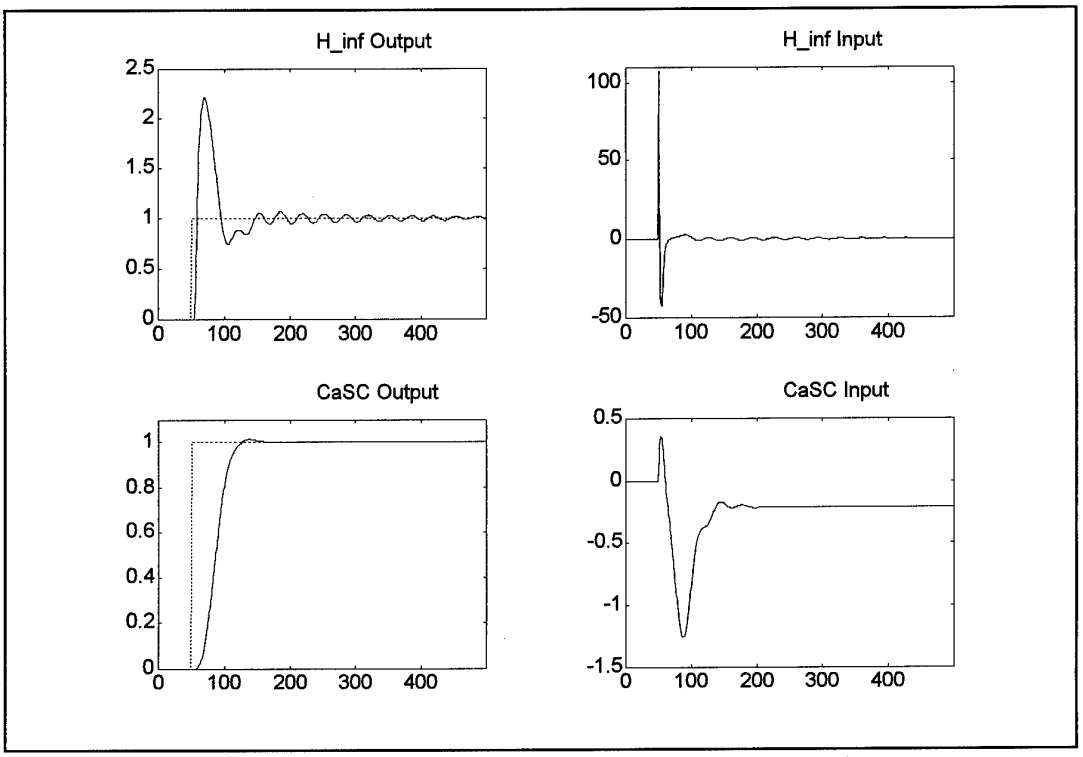


Figure 5.8 Tokamak Application - Simulated step response comparison

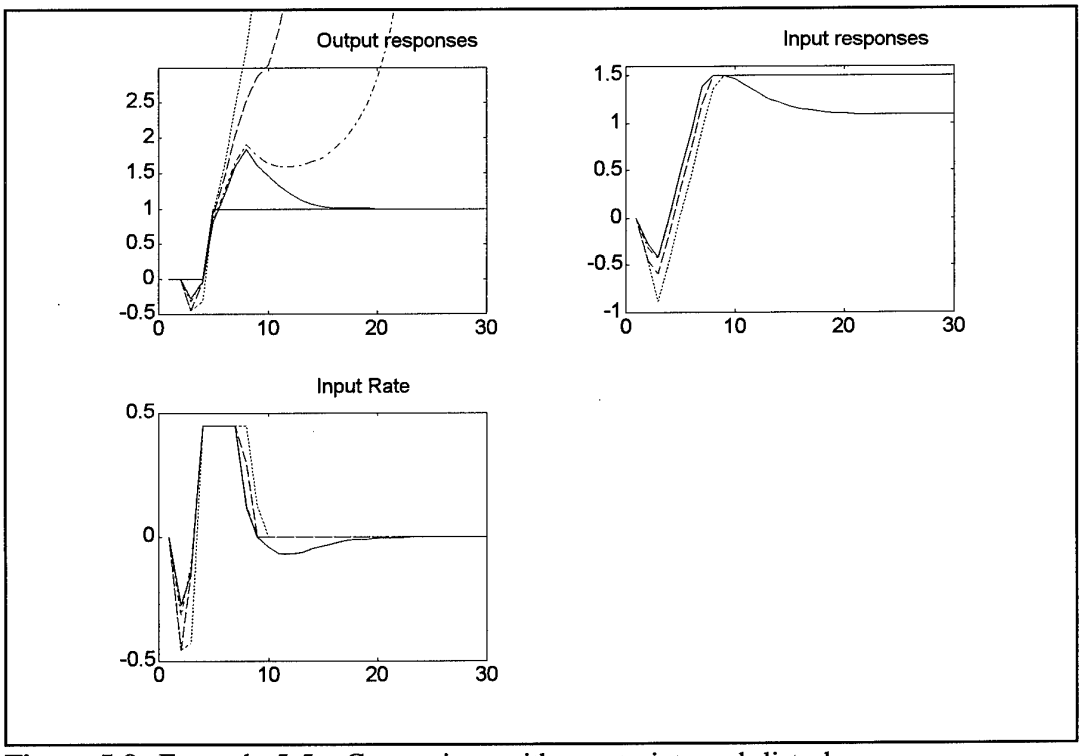


Figure 5.9 Example 5.5 - Comparison with constraints and disturbance

Chapter 6

Stability Results for Systems Subject to Disturbances

Disturbances are omnipresent and, for systems subject to input constraints, can drive control algorithms into infeasibility and instability. This problem has attracted little research effort, despite its significant practical importance. Previous chapters consider the disturbance-free case; the final purpose of this thesis is to address the issue of constrained predictive control in the more realistic setting of systems which are subject to bounded disturbances. In Section 6.1, we consider a description of how past and future disturbances enter into the system prediction equations; then, in Section 6.2, we show that the explicit stability results of Section 3.1 carry over to the more general case of systems which are subject to disturbances.

While the work of Section 6.2 gives explicit stability conditions, it is restricted to systems with at most one unstable pole, and does not lead to suitable algorithms because it applies to infinite horizons only. In Section 6.3, we develop necessary and sufficient limits on the size of inputs required to reject all possible disturbances and then modify the constraint limits of MCSGPC to derive an algorithm with guaranteed stability and asymptotic tracking. The application of all results in the chapter are illustrated by

means of numerical examples.

6.1 Disturbances: the stabilizing loop and the prediction equations

Most systems are subject to disturbances and a usual model for accounting for these is:

$$y_t = \frac{z^{-1}b(z)}{a(z)}u_t + \frac{T(z)}{a(z)}\zeta_t; \quad \zeta_t = \frac{1}{\Delta}\xi_t \tag{6.1}$$

where the expected value of ξ_t is taken to be zero. $T(z)$ is a polynomial in z which for clarity here will be taken to be 1; the $T(z) \neq 1$ case is straightforward, but leads to more cumbersome prediction equations. Often the assumption is also made that ξ_t is white noise, but this implies that ζ_t is a random walk; in many practical situations it is more realistic to assume that ζ_t is bounded, and accordingly, here we shall assume that ζ_t satisfies an inequality of the form:

$$|\zeta_t| \leq d_t; \quad d_t \geq 0 \qquad \forall\ t \tag{6.2}$$

If large disturbances are rare, it may be judicious to choose d_t smaller so as to account for the more typical disturbances only. For ease of presentation and without loss of generality, we shall assume the disturbance bound d_t is time-invariant and equal to d.

In the absence of constraints, the introduction of disturbances does not affect the stability properties of GPC or SGPC, though it may be advisable to use some filtering to improve performance ([7], [19]). However, disturbances in the presence of constraints can have disastrous effects if explicit account is not taken of them during the constrained optimization stage. Uncatered for disturbances can lead to infeasibility and, in the case of open loop unstable systems, are likely to lead to instability.

6.1.1 Deadbeat disturbance rejection

The proof of CSGPC stability depends on the assumption that the predicted output reaches its steady-state within a finite horizon. This is also true with disturbances and implies that the control law must reject finite duration disturbances within finite time. Let $\xi(z)$ be the z-transform of such a future disturbance, let $\Delta u(z)$, $y(z)$ be the z-transforms of the future (predicted) signals for Δu and y, and as with SGPC, let $c(z)=c_t+c_{t+1}z^{-1}+\cdots+c_{t+n_c-1}z^{-n_c+1}+c_\infty z^{-n_c}/(1-z^{-1})$. Then, from eqn. (2.13) with eqn. (6.1), we may write:

$$\begin{aligned} y(z)&=y^{(1)}(z)+y^{(2)}(z); \qquad \Delta u(z)=\Delta u^{(1)}(z)+\Delta u^{(2)}(z);\\ y^{(1)}(z)&=b(z)c(z)+y_f(z); \qquad \Delta u^{(1)}(z)=\alpha(z)c(z)+\Delta u_f(z);\\ y^{(2)}(z)&=\frac{b(z)\Delta u^{(2)}(z)+\xi(z)}{\alpha(z)} \end{aligned} \tag{6.3}$$

where $\Delta u(z)$ has been decomposed into the sum of $\Delta u^{(1)}(z)$ and $\Delta u^{(2)}(z)$; the former takes account of everything but future disturbances, whereas the latter is the part of the control law that is available for rejecting $\xi(z)$. Accordingly, it is obvious that $y^{(1)}(z)$, $\Delta u^{(1)}(z)$ are the z-transforms of the predicted y and Δu signals defined in eqn. (2.13); in particular eqns. (6.3)c,d are the z-transform equivalents of eqns. (2.13)a,b. Eqn. (6.3)e follows directly from eqn. (6.1) for zero initial conditions; the actual initial conditions have been incorporated in (6.3)c,d. Rewriting (6.3)e, we have:

$$\alpha(z)y^{(2)}(z)+b(z)(-\Delta u^{(2)}(z))=\xi(z) \tag{6.4}$$

Now the stabilizing loop of Figure 2.1 results in the FLS type of relationships of eqns. (2.2)a,b and (6.3)c,d, and hence, $y^{(1)}$ $\Delta u^{(1)}$ settle within finite time (n_y and n_u respectively). Clearly if y, Δu themselves are to reach a steady-state within the set finite horizons, $y^{(2)}(z)$, $\Delta u^{(2)}(z)$ must be polynomials in z^{-1} for all ξ. From eqn. (6.4), this will only hold true for all $\xi(z)$ if:

$$y^{(2)}(z)=M(z)\xi(z);\quad \Delta u^{(2)}(z)=-z^{-1}N(z)\xi(z);\quad \alpha(z)M(z)+z^{-1}b(z)N(z)=1 \tag{6.5}$$

Eqn. (6.5)c is the same bezout identity as eqn. (2.4) and so shares the same family of solutions; however, since the solutions are not unique, $M(z)$, $N(z)$ and $M^{\#}(z)$, $N^{\#}(z)$ can be chosen to be different.

6.1.2 The prediction equations

Eqns. (6.5)a,b, with the definition of ζ (eqn. (6.1)b), imply the z-transform and prediction equations:

$$\begin{aligned} y^{(2)}(z)&=M(z)\big(\Delta(z)\zeta(z)-\zeta_t\big); & \Delta u^{(2)}(z)&=-z^{-1}N(z)\big(\Delta(z)\zeta(z)-\zeta_t\big);\\ \underset{\rightarrow}{y}^{(2)}&=[C_{\Delta M}\underset{\rightarrow}{\zeta}-v_M\zeta_t]; & \underset{\rightarrow}{\Delta u}^{(2)}&=[-C_{z^{-1}\Delta N}\underset{\rightarrow}{\zeta}+v_{z^{-1}N}\zeta_t] \end{aligned} \tag{6.6}$$

where v_M, $v_{z^{-1}N}$ are column vectors of dimension n_y, n_u and comprise the coefficients of $M(z)$, $z^{-1}N(z)$; elements of these vectors that come after the last coefficient of $M(z)$ or $z^{-1}N(z)$ are zero. Eqns. (6.6)c,d combine with the prediction equations for $\underset{\rightarrow}{y}^{(1)}$ and $\underset{\rightarrow}{\Delta u}^{(1)}$ of eqn. (2.13) to give:

$$\begin{aligned} \underset{\rightarrow}{y}&=\underset{\rightarrow}{y}^{(1)}+\underset{\rightarrow}{y}^{(2)}; & \underset{\rightarrow}{y}^{(1)}&=\Gamma_b\underset{\rightarrow}{c}+\theta_b c_\infty+\underset{\rightarrow}{y_f}; & \underset{\rightarrow}{y}^{(2)}&=[C_{\Delta M}\underset{\rightarrow}{\zeta}-v_M\zeta_t]\\ \underset{\rightarrow}{\Delta u}&=\underset{\rightarrow}{\Delta u}^{(1)}+\underset{\rightarrow}{\Delta u}^{(2)}; & \underset{\rightarrow}{\Delta u}^{(1)}&=\Gamma_\alpha\underset{\rightarrow}{c}+\theta_\alpha c_\infty+\underset{\rightarrow}{\Delta u_f}; & \underset{\rightarrow}{\Delta u}^{(2)}&=[-C_{z^{-1}\Delta N}\underset{\rightarrow}{\zeta}+v_{z^{-1}N}\zeta_t]\\ \underset{\rightarrow}{u}&=\underset{\rightarrow}{u}^{(1)}+\underset{\rightarrow}{u}^{(2)}; & \underset{\rightarrow}{u}^{(1)}&=\Gamma_a\underset{\rightarrow}{c}+\theta_a c_\infty+\underset{\rightarrow}{u_f}; & \underset{\rightarrow}{u}^{(2)}&=[-C_{z^{-1}N}\underset{\rightarrow}{\zeta}+C_\Delta^{-1}v_{z^{-1}N}\zeta_t] \end{aligned} \tag{6.7}$$

In the above, we have assumed that the current value of the disturbance, ζ_t, is unknown; hence, it has been included in $\underset{\rightarrow}{y}^{(2)}$, $\underset{\rightarrow}{\Delta u}^{(2)}$, and $\underset{\rightarrow}{u}^{(2)}$. Given a record of past inputs and outputs, the value of ζ_t can be calculated and therefore can be considered as known; correspondingly, eqn. (6.7) can be rewritten:

$$\begin{array}{lll} \underset{\rightarrow}{y}=\underset{\rightarrow}{y}^{(1)}+\underset{\rightarrow}{y}^{(2)}; & \underset{\rightarrow}{y}^{(1)}=\Gamma_b\underset{\rightarrow}{c}+\theta_b c_\infty+\underset{\rightarrow}{y}_f-\nu_M\zeta_t; & \underset{\rightarrow}{y}^{(2)}=[C_{\Delta M}\underset{\rightarrow}{\zeta}] \\ \underset{\rightarrow}{\Delta u}=\underset{\rightarrow}{\Delta u}^{(1)}+\underset{\rightarrow}{\Delta u}^{(2)}; & \underset{\rightarrow}{\Delta u}^{(1)}=\Gamma_\alpha\underset{\rightarrow}{c}+\theta_\alpha c_\infty+\underset{\rightarrow}{\Delta u}_f+\nu_{z^{-1}N}\zeta_t; & \underset{\rightarrow}{\Delta u}^{(2)}=[-C_{z^{-1}\Delta N}\underset{\rightarrow}{\zeta}] \\ \underset{\rightarrow}{u}=\underset{\rightarrow}{u}^{(1)}+\underset{\rightarrow}{u}^{(2)}; & \underset{\rightarrow}{u}^{(1)}=\Gamma_a\underset{\rightarrow}{c}+\theta_a c_\infty+\underset{\rightarrow}{u}_f+C_\Delta^{-1}\nu_{z^{-1}N}\zeta_t; & \underset{\rightarrow}{u}^{(2)}=[-C_{z^{-1}N}\underset{\rightarrow}{\zeta}] \end{array} \qquad \textbf{(6.8)}$$

Of these two sets of prediction equations, (6.8) is most likely to be used in practice, but for completeness (eg. to allow for transient behaviour where disturbance information may be poor), here we shall consider both the cases of ζ_t unknown and known.

Given that the future (and possibly the current) disturbance values of ξ are unknown and zero-mean, it is tempting to ignore $\underset{\rightarrow}{y}^{(2)}$, $\underset{\rightarrow}{\Delta u}^{(2)}$, and $\underset{\rightarrow}{u}^{(2)}$. This is common practice in predictive control schemes such as GPC (and SGPC) and leads to optimal results. In the presence of constraints, however, such a policy could be catastrophic: to optimize performance, constrained predictive control algorithms tend to drive the controls close to the constraint limits; hence current and future disturbance sequences can drive the controls outside the constraints, thereby rendering the problem infeasible and leading to instability.

6.2 Explicit stability conditions with disturbances

Here we build on the results of Section 3.1 by showing that for some cases, explicit results are available which provide necessary and sufficient stability conditions in the presence of bounded constraints. In section 6.2.1, under an assumption of norm-boundedness, we derive the *a posteriori* necessary and sufficient stability conditions in the presence of disturbances; and finally, in section 6.2.2, we push these conditions one step into the future and show how they can be used to avoid instability.

6.2.1 *A posteriori* conditions for stability with disturbances

This section presents the explicit *a posteriori* conditions needed to guarantee the existence of a stabilizing solution for a plant subject to input absolute and rate constraints (2.28) in the presence of additive output disturbances. We first state the general result and then focus on systems with just one unstable pole.

Theorem 6.1 Let $a(z)$, p_i, q_i, and Q be defined per Theorem 3.2. Also, let the plant output be subject to the disturbance of Section 6.1. Then at time t, the problem is feasible (and CSGPC is stable) if, and only if, for $n_u \to \infty$, there exists a constrained $\underset{\rightarrow}{u}$ which satisfies:

$$Q\underset{\rightarrow}{u} + D_{b(p)}^{-1} Q\underset{\rightarrow}{\zeta} = \boldsymbol{b}_u; \qquad \text{where} \qquad \boldsymbol{b}_u = D_{b(p)}^{-1} Q(H_a \underset{\leftarrow}{y} - H_b \underset{\leftarrow}{u}) \tag{6.9}$$

where $D_{b(p)}$ is as defined for Theorem 3.3.

Proof: Multiplying eqn. (6.1) times $a(z)$ and simulating forward in time, we may write:

$$\underset{\rightarrow}{y} = C_a^{-1}(C_b \underset{\rightarrow}{u} - H_a \underset{\leftarrow}{y} + H_b \underset{\leftarrow}{u} + \underset{\rightarrow}{\zeta}) = C_a^{-1} w \tag{6.10}$$

This implies, by the proof of Theorem 3.2, that for $\underset{\rightarrow}{y}$ to remain bounded, w must be orthogonal to the rows of Q, or:

$$Q\underset{\rightarrow}{u} = D_{b(p)}^{-1} Q(H_a \underset{\leftarrow}{y} - H_b \underset{\leftarrow}{u} - \underset{\rightarrow}{\zeta}) \tag{6.11}$$

Rearrangement of eqn. (6.11) completes the proof. □

Considering the special case of plants with only one unstable pole leads to the following lemma and theorem:

Lemma 6.1 Let $a(z)$ have only one unstable pole at $p=1/q$, and let the disturbance, ζ, of Section 6.1, be bounded such that $|\zeta_t| \leq d$ for all t. Then the effect of the disturbance on eqn. (6.9) is bounded such that:

$$\frac{-d}{(1-|q|)\,|b(p)|} \leq \frac{1}{b(p)} Q\underset{\rightarrow}{\zeta} \leq \frac{d}{(1-|q|)\,|b(p)|} \tag{6.12}$$

Proof: The vector of future ζ's which maximizes $Q\underset{\rightarrow}{\zeta}$ is of the form (for $p>1$) $[d,d,d,\ldots]$ or (for $p<-1$) $[d,-d,d,\ldots]$, which, when premultiplied with Q, yields $d/(1-|q|)$. All other disturbances will have effects inside the limits of inequality (6.12). □

Theorem 6.2 Let $a(z)$ have only one unstable pole at $p=1/q$. Also, let the system be subject to input absolute and rate constraints (2.28) and the disturbance, ζ, of Section 6.1, bounded such that $|\zeta_t| \leq d$ for all t. Then at time t, the necessary and sufficient condition for the guaranteed existence of a stabilizing solution (LTF) is:

$$[b_u]_{\min} + \frac{d}{(1-|q|)\,|b(p)|} < b_u < [b_u]_{\max} - \frac{d}{(1-|q|)\,|b(p)|} \tag{6.13}$$

where $[b_u]_{\max}$ and $[b_u]_{\min}$ are given in eqn. (3.17).

Proof: Eqn. (6.13) guarantees the existence of a stabilizing solution in the presence of the worst case disturbance of Lemma 6.1. If b_u lies in the interval above, there will always exist a $\underset{\rightarrow}{u}$ which satisfies eqn. (6.9) (implying stability) and goes to the desired steady-state value without violating the constraints. If it does not lie in the interval, there is at least one vector of future ζ's for which no stabilizing solution exists. □

Corollary 6.1 Eqn. (6.9) can be rewritten as:

$$Q\underset{\rightarrow}{\Delta u} + D_{\Delta(p)} D_{b(p)}^{-1} Q\underset{\rightarrow}{\xi} = \underset{\rightarrow}{b}_{\Delta u} \qquad \text{where} \qquad \underset{\rightarrow}{b}_{\Delta u} = D_{\Delta(p)} D_{b(p)}^{-1} Q(H_a \underset{\leftarrow}{y} - H_b \underset{\leftarrow}{u}) - u_{t-1}\mathbf{1} \tag{6.14}$$

and the interval of eqn. (6.13) will then be:

$$[b_{\Delta u}]_{\min} + \frac{(1-q)d}{(1-|q|)\,|b(p)|} < b_{\Delta u} < [b_{\Delta u}]_{\max} - \frac{(1-q)d}{(1-|q|)\,|b(p)|} \tag{6.15}$$

with $[b_{\Delta u}]_{\max}$ and $[b_{\Delta u}]_{\min}$ as given in eqn. (3.12) or (3.16).

Proof: Eqn. (2.15) can be written as $\underset{\rightarrow}{\Delta u} = C_{\Delta}\underset{\rightarrow}{u} - u_{t-1}\boldsymbol{e}_1$ where $\boldsymbol{e}_1$ is the first standard vector. Premultiplying this equation with Q and substituting eqn. (6.11) into the result gives:

$$Q\underset{\rightarrow}{\Delta u} = D_{b(p)}^{-1} D_{\Delta(p)} Q(H_a \underset{\leftarrow}{y} - H_b \underset{\leftarrow}{u} - \underset{\rightarrow}{\xi}) - u_{t-1}\mathbf{1} \tag{6.16}$$

where we note that $QC_{\Delta} = D_{\Delta(p)}Q$ and $Q\boldsymbol{e}_1 = \mathbf{1}$. Rearrangement of eqn. (6.16) gives eqn. (6.14). Eqn. (6.15) follows from the arguments given in the proof of Theorem 6.2. □

Example 6.1 Let the system of Example 3.2 be subject to input absolute and rate constraints (2.28) for which $U_o=0$, $U=0.1$, and $R=0.2$. In Figure 6.1, the system is subject to a step disturbance bounded by $d=0.1$ at $t=20$. Figure 6.1d shows $[b_u]_{\max}$ and $[b_u]_{\min}$ (dashed line), the interval of Theorem 6.3 (dotted line), and b_u (solid line). b_u stays inside the interval and the system remains stable. In Figure 6.2, the size of the step disturbance is increased to $d=0.14$. Figure 6.2d shows that b_u comes right to the edge of the interval, which is seen in Figure 6.2a to have greatly affected the performance of the system. In Figure 6.3, the disturbance is increased slightly ($d=0.141$), causing b_u to go outside the interval and the system to go unstable.

6.2.2 *A priori* conditions for stability with disturbances

Paralleling the development of Section 3.1.3, here we present bounds on the current control move by propagating the *a posteriori* condition one step into the future, thus preserving the existence of a stabilizing solution at the next time instant. We will also show how these bounds indicate an effective limit on the size of disturbances hitting a system, or show when actuators may be inadequate for a given system and disturbance level.

Theorem 6.3 Let $a(z)$ have only one unstable pole at $p=1/q$. Also, let the system be subject only to input absolute constraint (2.28)a and the bounded disturbance, ζ, of Theorem 6.2. Then, at time t, a stabilizing solution will be guaranteed to exist at the next time instant (ie. the problem will be LTF) if, and only if, u_t is chosen such that:

$$b_u - \frac{qU_o}{1-q} - \frac{|q|U}{1-|q|} + \frac{d}{(1-|q|)|b(p)|} < u_t < b_u - \frac{qU_o}{1-q} + \frac{|q|U}{1-|q|} - \frac{d}{(1-|q|)|b(p)|} \qquad \textbf{(6.17)}$$

Proof: Extracting u_t from eqn. (6.9) (written for just one unstable pole) gives:

$$u_t = b_u - qq^T \underset{\rightarrow t+1}{u} - \frac{1}{b(p)} q^T \underset{\rightarrow}{\zeta} \qquad \textbf{(6.18)}$$

The result follows from application of lemmata 3.6 and 6.1. If u_t lies in interval (6.17), then at the next time instant there will always exist a $\underset{\rightarrow}{u}$ which satisfies eqn. (6.9) (implying stability) without violating the constraints. If it does not lie in the interval, there is at least one vector of future ζ's for which no stabilizing solution will exist. □

Theorem 6.4 Let $a(z)$ have only one unstable pole at $p=1/q$. Also, let the system be subject only to input rate constraint (2.28)b and the bounded disturbance, ζ, of Theorem 6.2. Then, at time t, a stabilizing solution will be guaranteed to exist at the next time

instant if, and only if, Δu_t is chosen such that:

$$b_{\Delta u} - \frac{|q|R}{1-|q|} + \frac{(1-q)d}{(1-|q|)|b(p)|} < \Delta u_t < b_{\Delta u} + \frac{|q|R}{1-|q|} - \frac{(1-q)d}{(1-|q|)|b(p)|} \qquad \textbf{(6.19)}$$

Proof: Extracting Δu_t from eqn. (6.14) (written for just one unstable pole) gives:

$$\Delta u_t = b_{\Delta u} - q\underset{\rightarrow}{q}^T \underset{t+1}{\Delta \mathbf{u}} - \frac{1-q}{b(p)}\underset{\rightarrow}{q}^T \underset{\rightarrow}{\zeta} \qquad \textbf{(6.20)}$$

The result then follows from the same arguments given in the proof of Theorem 6.3, except one uses Lemma 3.7 rather than Lemma 3.6. □

Example 6.2 Here, we again use the system and input constraint limits of Example 6.1, but note that R is equal to the width of the absolute constraint interval and thus, satisfaction of absolute constraints implies satisfaction of rate constraints; therefore the system can be considered to be subject to absolute constraints only. If the system encounters a step disturbance of magnitude 0.13 at t=8, then CSGPC will be unstable. This example shows why and shows how the instability might be avoided. Figure 6.4c shows the control inputs and constraints as before, but adds interval (6.17) (dotted lines) for d=0.14. At t=8, u_t exceeds this interval and CSGPC then encounters a disturbance which it cannot stabilize. For Figure 6.5, CSGPC is given the added hard constraint that it must not exceed interval (6.17). At t=8, u_t is "clipped" to stay within the interval (Figure 6.5c), and the system not only avoids instability, but maintains excellent performance when it encounters the disturbance.

An analysis of these bounds in the presence of a disturbance which reaches a steady-state value indicates certain limits on either the maximum size of disturbances hitting the system, or the minimum constraints that can be placed on the control inputs or their

increments. Before presenting this result, we give two lemmata; the first examines the structure of $\boldsymbol{q}^T H_a$, $\boldsymbol{q}^T H_b$ and the second defines steady-state values which are the result of the straight forward application of algebra and thus are presented without proof:

Lemma 6.2 Let $P\{.\}$ be the operator which maps the vector of coefficients $\boldsymbol{f}$ to $f(z)$, then

$$P\{\boldsymbol{q}^T H_f\} = -p\frac{f(z)-f(p)}{1-pz^{-1}} \tag{6.21}$$

Proof: Applying the definitions of $\boldsymbol{q}^T$ and H_f to the P operator yields:

$$P\{\boldsymbol{q}^T H_f\} = P\left\{\begin{bmatrix}1 & q & q^2 & \cdots & q^{n_f-1}\end{bmatrix}\begin{bmatrix}f_1 & f_2 & \cdots & f_{n_f}\\ f_2 & \cdots & f_{n_f} & \\ \vdots & & & \\ f_{n_f} & & & \end{bmatrix}\right\} = \begin{bmatrix}(f_1+f_2q+\cdots+f_{n_f}q^{n_f-1})\\ +(f_2+\cdots+f_{n_f}q^{n_f-2})z^{-1}\\ \vdots\\ +f_{n_f}z^{-(n_f-1)}\end{bmatrix} \tag{6.22}$$

Then, multiplying and dividing by $-p$ gives:

$$P\{\boldsymbol{q}^T H_f\} = -p\begin{bmatrix}-(f_1q+f_2q^2+\cdots+f_{n_f}q^{n_f})\\ -(f_2q+\cdots+f_{n_f}q^{n_f-1})z^{-1}\\ \vdots\\ -f_{n_f}qz^{-(n_f-1)}\end{bmatrix} \tag{6.23}$$

and dividing and multiplying by $1-pz^{-1}$ gives:

$$P\{\boldsymbol{q}^T H_f\} = \frac{-p}{1-pz^{-1}}\begin{bmatrix}-(f_1q+f_2q^2+\cdots+f_{n_f}q^{n_f})\\ +(f_1+f_2q+\cdots+f_{n_f}q^{n_f-1})z^{-1}\\ -(f_2q+\cdots+f_{n_f}q^{n_f-1})z^{-1}\\ \vdots\\ +f_{n_f}z^{-n_f}\end{bmatrix} \tag{6.24}$$

and finally, substituting $f_0-f(p)=-(f_1q+f_2q^2+\cdots+f_{n_f}q^{n_f})$ and cancelling like terms gives:

$$P\{q^T H_f\} = \frac{-p}{1-pz^{-1}}(f_0 - f(p) + f_1 z^{-1} + \cdots + f_{n_f} z^{-n_f}) = -p\frac{f(z)-f(p)}{1-pz^{-1}} \qquad \textbf{(6.25)}$$

which completes the proof. □

Remark 6.1 q is the vector of coefficients of $1/(1-qz^{-1})$ truncated after n_u terms, and by [19], $q^T H_f = P^{-1}\{[(P\{q\})^* [z f(z)]_-]_-\}$, where $(.)^*$ converts negative powers of z to positive and $[.]_-$ extracts only the causal terms. $f(p)$ can then be seen as a factor associated with the truncation of anti-causal terms.

Lemma 6.3 Let $a(z)$ have only one unstable pole at $p=1/q$, and let the system be subject to the bounded disturbance, ζ, of Theorem 6.2, which reaches a steady-state value of ζ_{ss}. If the system reaches steady-state at the desired value, r_{ss}, then we have:

$$\begin{aligned} y_{ss} &= r_{ss}; \qquad u_{ss} = \frac{a(1)}{b(1)} r_{ss} - \frac{1}{b(1)}\zeta_{ss}; \qquad \Delta u_{ss} = 0 \\ b_{u_{ss}} &= \frac{\gamma(1)}{b(1)} r_{ss} + \frac{\beta(1)}{b(1)b(p)}\zeta_{ss}; \qquad b_{\Delta u_{ss}} = \frac{1}{b(p)}\zeta_{ss} \end{aligned} \qquad \textbf{(6.26)}$$

where

$$\begin{aligned} \gamma(z) &\equiv P\{QH_a\} = \frac{-p\,a(z)}{1-pz^{-1}} = -p\,a^-(z); \qquad \gamma(1) = \frac{a(1)}{1-q} \\ \beta(z) &\equiv P\{QH_b\} = \frac{-p[b(z)-b(p)]}{1-pz^{-1}}; \qquad \beta(1) = \frac{b(1)-b(p)}{1-q} \end{aligned} \qquad \textbf{(6.27)}$$

Corollary 6.2 Let $a(z)$ have only one unstable pole at $p=1/q$, and let the system be subject only to input absolute constraint (2.28)a and the bounded disturbance, ζ, of Theorem 6.2 which reaches a steady-state value of $\zeta_{ss}=\pm d$. Then a necessary condition for the guaranteed existence of a stabilizing solution which tracks the commanded reference, r_{ss}, is:

$$\left|\frac{a(1)}{b(1)}r_{ss} - U_o\right| < \frac{1-q}{1-|q|}U - \left|\frac{(1-q)b(1) \pm (1-|q|)[b(1)-q\,b(p)]}{|q|(1-|q|)b(1)b(p)}\right|d \qquad \textbf{(6.28)}$$

Proof: Rewrite interval (6.17) as:

$$\left|u_t - b_{u_t} + \frac{qU_o}{1-q}\right| < \frac{|q|U}{1-|q|} - \frac{d}{(1-|q|)|b(p)|} \qquad \textbf{(6.29)}$$

Subtract eqn. (6.26)d from eqn. (6.26)b to get:

$$u_{ss} - b_{u_{ss}} = \frac{a(1)-\gamma(1)}{b(1)}r_{ss} - \frac{b(p)+\beta(1)}{b(1)b(p)}\zeta_{ss} \qquad \textbf{(6.30)}$$

then substitute eqns. (6.27)b & d to get:

$$u_{ss} - b_{u_{ss}} = \frac{1}{(1-q)}\left[\frac{q\,b(p)-b(1)}{b(p)b(1)}\zeta_{ss} - \frac{q\,a(1)}{b(1)}r_{ss}\right] \qquad \textbf{(6.31)}$$

Finally, substitute this result into interval (6.29) and set $\zeta_{ss}=\pm d$. A reference set-point which violates this inequality will have a steady-state input control, u_{ss}, which lies outside interval (6.17). If the control is allowed to reach this value, then there is at least one vector of future ζ's for which no stabilizing solution will exist. If u_t is constrained to lie inside interval (6.17), then the system will have a steady-state offset. □

Remark 6.2 Corollary 6.2 provides a hard limit on the size of reference inputs in the presence of bounded disturbances. Larger disturbances will either further limit reference inputs, or require wider constraints.

Corollary 6.3 Let $a(z)$ have only one unstable pole at $p=1/q$, and let the system be subject only to input rate constraint (2.28)b and the bounded disturbance, ζ, of Theorem 6.3 which reaches a steady-state value of $\zeta_{ss}=-d$. Then a necessary condition for the guaranteed existence of a stabilizing solution is:

$$\frac{d}{R} < \frac{|qb(p)|}{(1-|q|)+(1-q)} \tag{6.32}$$

Proof: Rewrite interval (6.19) as:

$$\left|\Delta u_t - b_{\Delta u_t}\right| < \frac{|q|R}{1-|q|} - \frac{(1-q)d}{(1-|q|)|b(p)|} \tag{6.33}$$

then substitute eqns. (6.26)c & e and ζ_{ss}=-d. If the ratio of the disturbance bound to the rate constraint bound violates this inequality, then interval (6.19) will not include zero. If the control increment is allowed to settle at zero, then there is at least one vector of future ζ's for which no stabilizing solution will exist. If Δu_t is constrained to lie inside interval (6.19), then the system will never reach steady-state. □

Remark 6.3 Corollary 6.3 provides a hard relationship between the bounds on disturbances and input rate constraints. Larger disturbances will require wider constraints.

Corollaries 6.2 and 6.3 address only steady-state conditions and are therefore necessary, but not sufficient for stability and asymptotic tracking. Transient conditions and non steady-state disturbances are not addressed.

Necessary and sufficient conditions for guaranteed stability in the presence of disturbances have been given, but are limited to the case of plants with at most one unstable pole and are based on infinite horizon conditions and thus do not lead to a predictive algorithm that retains asymptotic tracking in the presence of disturbances. Instead, in the following section, we remedy this by focusing attention on MCSGPC. We develop necessary and sufficient feasibility conditions and use these to develop a non-conservative extension to MCSGPC which copes with disturbances and has

guaranteed stability and asymptotic tracking.

6.3 Adding disturbance borders to MCSGPC

MCSGPC, as defined in Algorithm 4.4, is sensitive to disturbances; this can lead to infeasibility and instability. The purpose of this section is to take systematic account of disturbances, derive feasibility conditions, and extend MCSGPC so that it retains its stability and asymptotic tracking properties in the presence of bounded disturbances. The basic idea is to determine and reserve the minimum control authority required to reject all possible future disturbances. The implementation of this idea is effected by imposing input constraints which are tighter than those dictated by physical limits; the difference between the physical limits and imposed input constraints represents the control authority that needs to be reserved to maintain the feasibility of the constrained optimization problem and hence to guarantee the stability of the relevant predictive control algorithm.

Given a set of initial conditions and input constraints, the prediction equations of Section 6.1 are used to develop bounds on allowable future and current inputs such that the system can still be stabilised in the event of worst case future disturbance signals. Some freedom does exist in the determination of these bounds, and in the final subsection, this freedom is used to give less restrictive bounds on the inputs.

6.3.1 Necessary and sufficient MCSGPC feasibility conditions

To guarantee stability for systems subject to bounded disturbances, two sets of conditions must be derived and enforced: i) conditions under which stable input/output predictions do not violate constraints given any possible future disturbance sequence (we will term

these *a posteriori* conditions) and ii) conditions under which, given the effects of the disturbance that hits the system at the current time instant, the chosen prediction sequence will again be feasible at the following time instant (*a priori* conditions).

6.3.1.1 *A posteriori* conditions

Eqns. (6.7) and (6.8) contain terms which are unknown and cannot be explicitly accounted for; hence during optimization, one is effectively choosing $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}$, $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}$, not $\underset{\rightarrow}{\Delta \boldsymbol{u}}$, $\underset{\rightarrow}{\boldsymbol{u}}$. However, if $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}$, $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}$ are chosen subject to input absolute and rate constraints (2.28), then optimization can cause their elements to assume values close or equal to the constraint limits so that, for certain disturbance sequences, the actual predicted values of $\underset{\rightarrow}{\Delta \boldsymbol{u}}$, $\underset{\rightarrow}{\boldsymbol{u}}$ will violate the constraints. This would lead to infeasibility, causing $\underset{\rightarrow}{\boldsymbol{y}}$ not to behave as expected, and would thus invalidate the proofs of stability and asymptotic tracking. To prevent this, it is clear that the elements of $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}$, $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}$ must be subjected to tighter constraints:

$$\begin{matrix} |\Delta u^{(1)}_{t+i-1}| \le R - R^{\#}_i \\ |u^{(1)}_{t+i-1} - U_o| \le U - U^{\#}_i \end{matrix} \quad i=1,2,\ldots,n_u \quad \Leftrightarrow \quad \begin{matrix} -R\mathbf{1} + \boldsymbol{R}^{\#} \le \underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)} \le R\mathbf{1} - \boldsymbol{R}^{\#} \\ -U\mathbf{1} + \boldsymbol{U}^{\#} \le \underset{\rightarrow}{\boldsymbol{u}}^{(1)} - U_o\mathbf{1} \le U\mathbf{1} - \boldsymbol{U}^{\#} \end{matrix} \tag{6.34}$$

where $R^{\#}_i$ and $U^{\#}_i$, the elements of the vectors, $\boldsymbol{R}^{\#}$ and $\boldsymbol{U}^{\#}$, depend on eqns. (6.7)f,i or (6.8)f,i and thus on the values of unknown disturbances; henceforth, $\boldsymbol{R}^{\#}$ and $\boldsymbol{U}^{\#}$, the vectors of the amounts by which the actual constraints are tightened, will be called "borders". Clearly for guaranteed feasibility, the borders must allow for all possible unknown disturbances; this involves the determination of the worst case disturbance signal.

Theorem 6.5 (*A posteriori* feasibility conditions) For a given $N(z)=N_0+N_1z^{-1}+\ldots+N_kz^{-k}$

satisfying bezout identity (6.5)c, MCSGPC will give feasible $\underset{\rightarrow}{\Delta u}$ and $\underset{\rightarrow}{u}$ over all possible unknown disturbances if $\underset{\rightarrow}{\Delta u}^{(1)}$ and $\underset{\rightarrow}{u}^{(1)}$ of eqns. (6.7) (for ζ_t unknown) or (6.8) (for ζ_t known) are chosen to satisfy eqn. (6.34) for:

$$\zeta_t \text{ unknown:} \quad \begin{matrix} R_1^{\#}=0, & R_i^{\#}=d(|N_{i-2}|+|N_{i-2}-N_{i-3}|+|N_{i-3}-N_{i-4}|+\ldots+|N_0|), \\ U_1^{\#}=0, & U_i^{\#}=d(|N_{i-2}+N_{i-3}+\ldots+N_0|+|N_{i-2}|+|N_{i-3}|+\ldots+|N_0|), \end{matrix} \tag{6.35}$$

$$\zeta_t \text{ known:} \quad \begin{matrix} R_1^{\#}=0, & R_i^{\#}=d(|N_{i-2}-N_{i-3}|+|N_{i-3}-N_{i-4}|+\ldots+|N_0|), \\ U_1^{\#}=0, & U_i^{\#}=d(|N_{i-2}|+|N_{i-3}|+\ldots+|N_0|), \end{matrix} \quad i=2,3,\ldots,n_u \tag{6.36}$$

Furthermore, for the given $N(z)$, eqns. (6.35),(6.36) define the smallest borders for which feasibility can be guaranteed.

Proof: Eqn. (6.35) gives least upper bounds on the size of the elements of $\underset{\rightarrow}{\Delta u}^{(2)}, \underset{\rightarrow}{u}^{(2)}$ of eqn. (6.7):

$$\begin{aligned} -\boldsymbol{R}^{\#}=d\left[\boldsymbol{v}_{z^{-1}N} \vdots C_{z^{-1}\Delta N}\right]^{+}\mathbf{1} \le \underset{\rightarrow}{\Delta \boldsymbol{u}}^{(2)}=[-C_{z^{-1}\Delta N}\underset{\rightarrow}{\zeta}+\boldsymbol{v}_{z^{-1}N}\zeta_t]=\left[\boldsymbol{v}_{z^{-1}N} \vdots C_{z^{-1}\Delta N}\right]\left[\zeta_t \vdots -\underset{\rightarrow}{\zeta}^{T}\right]^{T} \\ \le d\left[\boldsymbol{v}_{z^{-1}N} \vdots C_{z^{-1}\Delta N}\right]^{+}\mathbf{1}=\boldsymbol{R}^{\#} \\ -\boldsymbol{U}^{\#}=d\left[C_{\Delta}^{-1}\boldsymbol{v}_{z^{-1}N} \vdots C_{z^{-1}N}\right]^{+}\mathbf{1} \le \underset{\rightarrow}{\boldsymbol{u}}^{(2)}=[-C_{z^{-1}N}\underset{\rightarrow}{\zeta}+C_{\Delta}^{-1}\boldsymbol{v}_{z^{-1}N}\zeta_t]=\left[C_{\Delta}^{-1}\boldsymbol{v}_{z^{-1}N} \vdots C_{z^{-1}N}\right]\left[\zeta_t \vdots -\underset{\rightarrow}{\zeta}^{T}\right]^{T} \\ \le d\left[C_{\Delta}^{-1}\boldsymbol{v}_{z^{-1}N} \vdots C_{z^{-1}N}\right]^{+}\mathbf{1}=\boldsymbol{U}^{\#} \end{aligned} \tag{6.37}$$

Where $[.]^{+}$ implies a matrix of absolute values. They are obtained for disturbance sequences whose instantaneous values are $\pm d$, the sign being chosen so as to match that of the corresponding coefficients of: $C_{z^{-1}\Delta N}$ and $\boldsymbol{v}_{z^{-1}N}$ for $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(2)}$; and $C_{z^{-1}N}$ and $C_{\Delta}^{-1}\boldsymbol{v}_{z^{-1}N}$ for $\underset{\rightarrow}{\boldsymbol{u}}^{(2)}$. The first term of the expression for $U_i^{\#}$ in eqn. (6.35)b appears because C_{Δ}^{-1} is a lower triangular matrix of 1's. Eqn. (6.36) is obtained in a similar manner. □

As a consequence of Theorem 6.5, a necessary condition for the stability of MCSGPC in the presence of disturbances, is that constraints (2.28) be replaced by (6.34) with the elements of $\boldsymbol{R}^{\#}$ and $\boldsymbol{U}^{\#}$ as given by eqns. (6.35) or (6.36). Clearly, the cost to be

optimized must be based on the expected disturbance, and as ξ is zero mean, the expected value of $\underset{\rightarrow}{y}^{(2)}$, $\Delta\underset{\rightarrow}{u}^{(2)}$ will be $\mathbf{0}$ and thus we still use prediction eqns. (2.13)a,b or (4.1)a,b rather than (6.7)a,d or (6.8)a,d for the calculation of J. Hence, for the time being MCSGPC shall be modified by replacing constraints (2.28) with (6.34)-(6.36), but in all other respects will be the same as Algorithm 4.4.

In the discussion below, which concerns the derivation of *a priori* conditions, we need to consider the implications of current choices of $\underset{\rightarrow}{c}$ on feasibility at the next time instant; in this context, the following corollary and definitions prove useful.

Corollary 6.4 The elements of the vector $\boldsymbol{R}^{\#}$ of Theorem 6.5, taken in order, form a monotonically increasing sequence. The same applies to the elements of $\boldsymbol{U}^{\#}$.

Proof: From the application of the triangle inequality we have that $|N_{i-2}-N_{i-1}|+|N_{i-1}|-|N_{i-2}|$ is positive; however this quantity is equal to $R_{i+1}^{\#}-R_i^{\#}$, which establishes the monotonically increasing nature of the elements of $\boldsymbol{R}^{\#}$ for the case of unknown/unmeasurable disturbances. The proof for known disturbances is the same, as is the proof for the elements of $\boldsymbol{U}^{\#}$. □

Definition 6.1: Let $\Delta\underset{\rightarrow}{u}^{(1)}(t)$ and $\Delta\underset{\rightarrow}{u}^{(1)}(t+1)$ denote the vector $\Delta\underset{\rightarrow}{u}^{(1)}$ computed at t, and at $t+1$, and let $\Delta\underset{\rightarrow}{u}^{(1)}(t+1\,|\,t)$ denote $\Delta\underset{\rightarrow}{u}^{(1)}$ at $t+1$, given the information known at t and given the predicted control law computed at t; thus if the input horizon for $\Delta\underset{\rightarrow}{u}^{(1)}(t)$ were increased by 1, then $\Delta\underset{\rightarrow}{u}^{(1)}(t+1\,|\,t)$ (with the original horizon) would comprise all, but the first element. The corresponding definitions will apply to $\underset{\rightarrow}{u}^{(1)}$.

6.3.1.2 *A priori* conditions

The conditions of Theorem 6.5 are necessary for guaranteeing MCSGPC stability; they ensure that predicted inputs will not violate the actual constraints for all possible future disturbances. However, they are in fact *a posteriori* conditions in that they stipulate constraints on $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}$, $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}$ given past records; depending on earlier control moves and current disturbances, it may or may not be possible to satisfy these constraints at the next instant. Another way of viewing the problem is this: in the absence of disturbances, STF at t guarantees STF at $t+1$ (and any subsequent time). Indeed (for no set-point change), all one has to do is continue using the future inputs computed at t; by Definition 6.1, this would be accomplished by selecting $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1)=\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1\,|\,t)$. However, with disturbances, this will not be enough because at $t+1$, ζ_t and ζ_{t+1} come into play, and as a result, a vector, say $f_u(\zeta_t,\zeta_{t+1})$, will be added onto $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1)$ which is not accounted for by $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1\,|\,t)$; a similar vector will effect the future control increments. The limits of (6.34)-(6.36) will be reapplied at $t+1$ and will have effectively shifted one step in time; given the monotonically increasing nature of the elements of $\boldsymbol{R}^{\#}$ and $\boldsymbol{U}^{\#}$ (Corollary 6.4), this time shift will make more control authority available to $\Delta u_i^{(1)}(t+1)$ and $u_i^{(1)}(t+1)$ than was available to $\Delta u_{i+1}^{(1)}(t)$ and $u_{i+1}^{(1)}(t)$. However this may not be enough to prevent STIF at $t+1$ if $f(\zeta_t,\zeta_{t+1})$ exceeds the size of the increase in control authority. For example, compare the authority available to $\Delta u_2^{(1)}(t)$ and $\Delta u_1^{(1)}(t+1)$, the second element of $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t)$ and the first element of $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1)$; the upper bounds for these control increments are given by R-$R_2^{\#}$, and R-$R_1^{\#}$, respectively and thus indicate an increase in authority of $R_2^{\#}$-$R_1^{\#}$ due to the shift in the upper bounds and also an equal amount due to the shift in the lower bounds. The argument here is that if the modulus of the first element of $f_{\Delta u}(\zeta_t,\zeta_{t+1})$ is greater than $R_2^{\#}$-$R_1^{\#}$ then, depending on the signs of ζ_t, ζ_{t+1} and how close

$\Delta u_1^{(1)}(t+1|t)$ is to its limit of R-$R_2^{\#}$ or $-R+R_2^{\#}$, MCSGPC may become infeasible at $t+1$ even though it was feasible at t. To prevent such a problem from occurring, one needs *a priori* feasibility conditions which not only ensure that predicted inputs will not violate the actual constraints for all possible future disturbances, but also ensure that MCSGPC (with the borders) will remain STF at the next instant. To that end, we: i) derive expressions for $f_{\Delta u}$ and f_u; ii) give the relationship between $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}(t+1)$, $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1)$ and $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}(t+1|t)$, $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1|t)$; iii) redefine borders which are at least as large as those of Theorem 6.5, but also, at the next sampling instant, release enough incremental control authority to cater for the worst case $f_{\Delta u}$ and f_u; and iv) use these new borders to state *a priori* feasibility conditions.

Lemma 6.4 Let $[c_t, c_{t+1}, \ldots, c_{t+n_c-1}, c_\infty, c_\infty, \ldots]$ be the optimal sequence of c values computed at t, and let the same values (except for the first) be used again at the next sampling instant. Then the difference that ζ_{t+1} and, if unknown, ζ_t cause in $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}(t+1)$ and $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1)$ is:

$$\zeta_t \text{ unknown:} \quad f_{\Delta u}(\zeta_t, \zeta_{t+1}) = -\boldsymbol{v}_N(\zeta_{t+1} - \zeta_t); \quad f_u(\zeta_t, \zeta_{t+1}) = -C_\Delta^{-1}\boldsymbol{v}_N(\zeta_{t+1} - \zeta_t) \tag{6.38}$$

$$\zeta_t \text{ known:} \quad f_{\Delta u}(\zeta_{t+1}) = -\boldsymbol{v}_{\Delta N}\zeta_{t+1}; \quad f_u(\zeta_{t+1}) = -\boldsymbol{v}_N\zeta_{t+1} \tag{6.39}$$

Proof: We require a comparison between the vectors $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}(t+1)$ and $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1)$ and the vectors $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}(t+1|t)$ and $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1|t)$. Advancing (6.7) and (6.8) forward by one step, it is easy to see that $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}(t+1|t)$ and $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1|t)$ have the same form as $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}(t+1)$ and $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1)$ for ζ_t, $\zeta_{t+1}=0$. However, non-zero ζ_t and ζ_{t+1} change this, since the first element of $\underset{\leftarrow}{\boldsymbol{y}}$ at $t+1$ is y_{t+1}. In particular, the first element of $\underset{\rightarrow}{\boldsymbol{y}}^{(2)}$ at t from eqn. (6.7)c is given by $M_0(\zeta_{t+1}-\zeta_t)$, and this term appears in y_{t+1} and therefore will also appear in

$\underset{\rightarrow}{\Delta u}^{(1)}(t+1)$ and $\underset{\rightarrow}{u}^{(1)}(t+1)$ (through $\underset{\leftarrow}{y}$) as:

ζ_t unknown:
$$\begin{aligned} f_{\Delta u}(\zeta_t,\zeta_{t+1}) &= -(P_3)_1 M_0(\zeta_{t+1}-\zeta_t) = -v_N(\zeta_{t+1}-\zeta_t) \\ f_u(\zeta_t,\zeta_{t+1}) &= -C_\Delta^{-1}(P_3)_1 M_0(\zeta_{t+1}-\zeta_t) = -C_\Delta^{-1} v_N(\zeta_{t+1}-\zeta_t) \end{aligned} \tag{6.40}$$

ζ_t known:
$$\begin{aligned} f_{\Delta u}(\zeta_{t+1}) &= (v_{z^{-1}N}-(P_3)_1 M_0)\zeta_{t+1} = (v_{z^{-1}N}-v_N)\zeta_{t+1} = -v_{\Delta N}\zeta_{t+1} \\ f_u(\zeta_{t+1}) &= C_\Delta^{-1}(v_{z^{-1}N}-(P_3)_1 M_0))\zeta_{t+1} = C_\Delta^{-1}(v_{z^{-1}N}-v_N))\zeta_{t+1} = -C_\Delta^{-1} v_{\Delta N}\zeta_{t+1} = -v_N\zeta_{t+1} \end{aligned} \tag{6.41}$$

where $(P_3)_1$ is the first column vector of P_3; use has been made of the fact that $M_0=1$ and that $(P_3)_1=v_N$. The z^{-1} in $v_{z^{-1}N}$ has the effect of shifting the elements of v_N down by one place, thus $v_{z^{-1}N}-v_N=-v_{\Delta N}$; the effect of C_Δ^{-1} on the other hand is to cancel the "Δ" from the subscript of $v_{\Delta N}$, hence establishing the last equality in eqn. (6.41)b. □

Lemma 6.5 The vectors $\underset{\rightarrow}{\Delta u}^{(1)}(t+1)$ and $\underset{\rightarrow}{u}^{(1)}(t+1)$ can be written as

$$\underset{\rightarrow}{\Delta u}^{(1)}(t+1) = \underset{\rightarrow}{\Delta u}^{(1)}(t+1\,|\,t) + \left[\Gamma_\alpha \quad \theta_\alpha\right]x + f_{\Delta u}; \qquad \underset{\rightarrow}{u}^{(1)}(t+1) = \underset{\rightarrow}{u}^{(1)}(t+1\,|\,t) + \left[\Gamma_a \quad \theta_a\right]x + f_u \tag{6.42}$$

where x is an arbitrary vector of conformal dimensions.

Proof: By Lemma 6.3, eqn. (6.42) holds true for $x=0$ if the predicted control law computed at t were to be continued at $t+1$. However at $t+1$, one is at liberty to change the current and future values of c, namely one can change the vector $[\underset{\rightarrow}{c}^T \;\; c_\infty]^T$, say by a vector x. The effect of x on $\underset{\rightarrow}{\Delta u}^{(1)}$ and $\underset{\rightarrow}{u}^{(1)}$, as dictated by eqns. (6.7),(6.8), is that given by the 2nd term on the RHS of eqns. (6.42)a,b. □

Lemma 6.6 For a given $N(z)$ (satisfying (6.5)c), let $R_i^\#$, $U_i^\#$ be defined as

ζ_t unknown: $R_1^\#=0$, $R_i^\#=R_{i-1}^\#+2d|N_{i-2}|$; $U_1^\#=0$, $U_i^\#=U_{i-1}^\#+2d|N_0+\ldots+N_{i-2}|$ (6.43)

ζ_t known: $R_1^\#=0$, $R_i^\#=R_{i-1}^\#+d|N_{i-2}-N_{i-3}|$; $U_1^\#=0$, $U_i^\#=U_{i-1}^\#+d|N_{i-2}|$ $i=2,\ldots,n_u$ (6.44)

then such $R_i^\#$ and $U_i^\#$ are greater than or equal to the corresponding borders of Theorem 6.5, and the differences, $R_i^\#-R_{i-1}^\#$ and $U_i^\#-U_{i-1}^\#$, define least upper bounds for the moduli

of the $(i\text{-}1)^{\text{th}}$ element of the corresponding $\boldsymbol{f}_{\Delta u}$ and $\boldsymbol{f}_u$ over all possible ζ_t and ζ_{t+1}.

Proof: The borders of eqn. (6.44) and (6.36) are precisely the same, whereas those of (6.43) are greater than or equal to those (6.35); this follows from a straightforward application of the triangle inequality. Finally the differences $R_i^{\#}\text{-}R_{i\text{-}1}^{\#}$ and $U_i^{\#}\text{-}U_{i\text{-}1}^{\#}$ define least upper bounds on the size of the elements of $\boldsymbol{f}_{\Delta u}$ and $\boldsymbol{f}_u$ because the largest values that $|\zeta_{t+1}\text{-}\zeta_t|$ and $|\zeta_{t+1}|$ can assume are $2d$ and d. □

Henceforth, we shall use the borders of eqns. (6.43),(6.44) in place of those of (6.35),(6.36), and by MCSGPC, we shall refer to the following algorithm:

Algorithm 6.1 (MCSGPC with disturbances)

This is identical to Algorithm 4.4 with constraints (2.28) replaced by (6.34) with $R_i^{\#}$ and $U_i^{\#}$ as given in eqns. (6.43) or (6.44).

Theorem 6.6 (*A priori* feasibility conditions) Given STF at t, MCSGPC will be STF at $t+i$ for all $i>1$ and will give feasible $\underset{\rightarrow}{\Delta \boldsymbol{u}}$ and $\underset{\rightarrow}{\boldsymbol{u}}$ over all possible unknown disturbances. Furthermore, for a given $N(z)$ (satisfying (6.5)c), the borders of Lemma 6.5 define the smallest borders for which continued STF can be guaranteed for all allowable future disturbances.

Proof: First we prove sufficiency, namely that the borders of Algorithm 6.1 guarantee that if MCSGPC is STF at t, it will be STF at $t+1$ and hence at all subsequent times and will give feasible $\underset{\rightarrow}{\Delta \boldsymbol{u}}$ and $\underset{\rightarrow}{\boldsymbol{u}}$. Next we establish necessity, by showing that STF cannot be guaranteed for smaller borders.

(Sufficiency) The assumption of STF at t implies that $\underset{\rightarrow}{\Delta u}^{(1)}(t)$, $\underset{\rightarrow}{u}^{(1)}(t)$ which satisfy constraints (6.34)c,d with $\boldsymbol{R}^{\#}$, $\boldsymbol{U}^{\#}$ as per Lemma 6.5 exist and therefore that MCSGPC will choose such vectors. By Lemma 6.5, we also have that the borders of Algorithm 6.1 are greater than or equal to those of eqns. (6.35),(6.36) and hence by Theorem 6.5, $\underset{\rightarrow}{\Delta u}(t)$, $\underset{\rightarrow}{u}(t)$ will be feasible over all possible unknown disturbances. Furthermore, satisfaction of the constraint on $\underset{\rightarrow}{\Delta u}^{(1)}(t)$ ((6.34)c) implies that:

$$-R\mathbf{1}+\boldsymbol{R}^{\#}+(E\boldsymbol{R}^{\#}-\boldsymbol{R}^{\#}) = -R\mathbf{1}+E\boldsymbol{R}^{\#} \le \underset{\rightarrow}{\Delta u}^{(1)}(t+1\,|\,t) \le R\mathbf{1}-E\boldsymbol{R}^{\#} = R\mathbf{1}-\boldsymbol{R}^{\#}-(E\boldsymbol{R}^{\#}-\boldsymbol{R}^{\#}) \qquad \textbf{(6.45)}$$

where E is a matrix such that $E\boldsymbol{R}^{\#}$ is the vector comprising the 2^{nd}, 3^{rd},...,n_u^{th},n_u^{th} elements of $\boldsymbol{R}^{\#}$. Hence, from eqn. (6.42) for $\boldsymbol{x}=0$, we have:

$$-R\mathbf{1}+\boldsymbol{R}^{\#}-\{\boldsymbol{f}_{\Delta u}-(E\boldsymbol{R}^{\#}-\boldsymbol{R}^{\#})\} \le \underset{\rightarrow}{\Delta u}^{(1)}(t+1) \le R\mathbf{1}-\boldsymbol{R}^{\#}+\{\boldsymbol{f}_{\Delta u}-(E\boldsymbol{R}^{\#}-\boldsymbol{R}^{\#})\} \qquad \textbf{(6.46)}$$

However, by definition, $(E\boldsymbol{R}^{\#}-\boldsymbol{R}^{\#})_i = R_{i+1}^{\#}-R_i^{\#}$ for $i=1,\ldots,n_u-1$ and 0 for $i=n_u$, so by Lemma 6.5, the elements of the vector inside the curly brackets are non-positive; hence (for $\boldsymbol{x}=0$) $\underset{\rightarrow}{\Delta u}^{(1)}(t+1)$ satisfies constraints (6.34)c and MCSGPC is STF at $t+1$. Clearly, this argument carries over to $t+2$, $t+3$, etc, and similar arguments apply to $\underset{\rightarrow}{u}^{(1)}$.

(Necessity) The proof given above concerns sufficiency only because the extra degrees of freedom available at $t+1$, encapsulated in the vector $\boldsymbol{x}$, have not been deployed; $\boldsymbol{x}$ has been taken to be zero. Now we show that it is not possible to use $\boldsymbol{x}$ to counteract the effects of $\boldsymbol{f}_{\Delta u}$ and $\boldsymbol{f}_u$ with the view to enabling the use of smaller borders. First note that for the case of ζ_t known, the new borders (eqn. (6.44)) coincide with those of eqn. (6.36) in Theorem 6.5, and hence it is clear that smaller borders would lead to MCSGPC solutions which do not satisfy the *a posteriori* conditions; thus the borders of eqn. (6.44) cannot be made smaller. For the case of ζ_t unknown, $\boldsymbol{f}_{\Delta u}$ and $\boldsymbol{f}_u$ are unknown

at t, and their effects on $\Delta\underset{\rightarrow}{u}^{(1)}(t+1)$ and $\underset{\rightarrow}{u}^{(1)}(t+1)$, which define the borders at t, cannot be decreased by the deployment of a non-zero $\boldsymbol{x}$ at $t+1$. □

6.3.2 Stability properties of MCSGPC with disturbances

Here, we first give conditions on d (or R and U) which are necessary for the implementation of MCSGPC and then show that the algorithm: i) is guaranteed stable and ii) gives asymptotic tracking for disturbances which reach a steady-state value.

Corollary 6.5 For a given $N(z)$ (as defined by eqn. (6.5)c), STF can be achieved, and asymptotic tracking can be guaranteed, only if

$$\zeta_t \text{ unknown:} \quad d \le \min\{\frac{R}{2\sum_i |N_i|}, \quad \frac{U}{2\sum_i |\sum_{j=1} N_j|}, \quad \frac{[U-|\frac{a(1)}{b(1)}r-U_0|]}{2\sum_i |\sum_{j=1} N_j| + |\sum_i N_i|}\} \tag{6.47}$$

$$\zeta_t \text{ known:} \quad d \le \min\{\frac{R}{|N_0| + |N_1 - N_0| + \ldots + |-N_k|}, \quad \frac{U}{\sum_i |N_i|}, \quad \frac{[U-|\frac{a(1)}{b(1)}r-U_0|]}{\sum_i |N_i| + |\sum_i N_i|}\} \tag{6.48}$$

Proof: The denominators of the first two terms of the RHS of (6.47),(6.48) are equal to the last elements of $\boldsymbol{R}^{\#}$ and $\boldsymbol{U}^{\#}$, which are the largest. Hence, the implied inequalities ensure that the bounds of Algorithm 6.1 (inequality (6.34), with the borders of eqns. (6.43) or (6.44)) are non-overlapping. Clearly, if this were not so, STF would not be possible. The inequalities implied by the third term of the RHS of (6.47) and (6.48) are a consequence of eqns. (6.7)g-i and (6.8)g-i, and ensure that the desired steady-state value for u, $a(1)r/b(1)$, cannot become infeasible in the presence of disturbances; violation of this condition would contradict asymptotic tracking. □

An interesting corollary to this result is that rather than assess stability given R, U and d, one can use information on the size of disturbances and likely set-point changes to determine the minimum values of R and U for which stability can be guaranteed. Clearly, this has implication in the selection of actuators and can be used to financial advantage.

Lemma 6.7 Assume STF at some t (eg $t=0$), then Algorithm 6.1 gives bounded input bounded output (BIBO) stability for all disturbances satisfying condition (6.2).

Proof: The prediction equation for the actual $y(z)$ comprises the known $y^{(1)}(z)$ and unknown $y^{(2)}(z)$. It is easy to see from eqn. (6.7)c that as ζ is bounded, $y^{(2)}(z)$ will be bounded. $y^{(1)}(z)$ can be computed from the transfer function model which implies that $\alpha(z)y^{(1)}(z)+p_2(z)=b(z)\Delta u^{(1)}(z)+p_1(z)$ where $p_1(z)$ and $p_2(z)$ account for initial conditions. Rearranging this equation and writing the corresponding prediction equations, we get:

$$\alpha(z)y^{(1)}(z)=b(z)\Delta u^{(1)}(z)+p_1(z)-p_2(z) \quad \Rightarrow \quad \underset{\rightarrow}{C_\alpha y}^{(1)}=\underset{\rightarrow}{C_b \Delta u}^{(1)}+\boldsymbol{p}_1-\boldsymbol{p}_2;$$
$$\boldsymbol{p}_1=H_b \underset{\leftarrow}{\Delta \boldsymbol{u}}; \quad \boldsymbol{p}_2=H_\alpha \underset{\leftarrow}{\boldsymbol{y}}; \quad p_1(z)=[1,z^{-1},...,z^{-n},0,...,0]\boldsymbol{p}_1; \quad p_2(z)=[1,z^{-1},...,z^{-n},0,...,0]\boldsymbol{p}_2 \tag{6.49}$$

The proof is now by contradiction. Clearly, given the SGPC strategy and the assumption of STF at start up (which by Theorem 6.6 implies STF at all times), the predicted $y^{(1)}(z)$ reaches a fixed value, $b(1)c_\infty$, in n_y steps. Therefore, solving eqn. (6.49)a for $y^{(1)}(z)$, we conclude that $b(z)\Delta u^{(1)}(z)+p_1(z)-p_2(z)$ has $a(z)$ as a factor. Assume that $\underset{\leftarrow}{\boldsymbol{y}}$ is going unbounded; then, as H_α has full column rank, $p_2(z)$ goes unbounded. On the other hand, constraints (2.28) imply that Δu is bounded and therefore $p_2(z)$ will dominate over $b(z)\Delta u^{(1)}(z)+p_1(z)$. This in turn implies that $p_2(z)$ itself has $a(z)$ as a factor; but since $p_2(z)$ and $a(z)$ have the same degree, $p_2(z)$ must equal $\mu a(z)$, with μ an arbitrary scalar.

Rewriting this in matrix form gives $H_\alpha \underset{\leftarrow}{y} = \mu[a_0, a_1, ..., a_n, 0, ..., 0]^T$. H_α has full column rank, hence the solution for $\underset{\leftarrow}{y}$ (if it exists) is unique. Furthermore, it is easy to see that this unique solution is given by $\underset{\leftarrow}{y} = -\mu[1, ..., 1]^T$. By the assumption of unboundedness, μ must be unbounded and therefore $p_2(1)$ must be unbounded. However, by applying the final value theorem to (6.49)a, we have that the steady-state value of $y^{(1)}(z)$, $b(1)c_\infty$, equals $[b(1)\Delta u^{(1)}(1) + p_1(1) - p_2(1)]/a(1)$, which is finite. As $b(1)\Delta u^{(1)}(1)$, $p_1(1)$, and $a(1)$ are known to be finite, $p_2(1)$ must be finite. This establishes the contradiction, so $\underset{\leftarrow}{y}$ cannot go unbounded. The proof is completed by noting that $\underset{\leftarrow}{y}$ contains the actual values of the output. □

Lemma 6.8 Assume STF at some t (eg $t=0$), then providing the disturbance ζ reaches a steady-state value, Algorithm 6.1 will have guaranteed asymptotic tracking.

Proof: From Theorem 6.6, STF at t implies STF for all future time. Hence if ζ reaches steady-state, then MCSGPC will force c_∞ to converge to $r/b(1)$, thereby making CSGPC feasible and therefore causing Algorithm 6.1 to revert to CSGPC. Finally, the integral action of CSGPC will ensure that the output converges to the correct value. □

Theorem 6.7: Assume STF at some t (eg $t=0$), then Algorithm 6.1 is stable and, provided ζ reaches a steady-state value, will have guaranteed asymptotic tracking.

Proof: This is a natural consequence of Lemmata 6.6 and 6.7. □

In concluding this section, we point out that the borders of Theorems 6.5-7 are proportional to d, so that choosing d large would limit the control authority available for the optimization of performance. On the other hand choosing d small might not account

for all possible disturbances and would increase the likelihood of instability; the "optimal" choice for d is a matter of engineering judgement.

6.3.3 Utilizing available degrees of freedom

The previous development still has some freedom of choice; the first is in $N(z)$, and the second is in $\boldsymbol{x}$.

6.3.3.1 Optimization of the borders

The solutions $M^{\#}(z)$, $N^{\#}(z)$ of bezout identity (2.4) do not appear in eqn. (2.2) and so, in the absence of disturbances, do not effect the prediction equations and thus do not effect MCSGPC. Similarly, the solutions $M(z)$, $N(z)$ of eqn. (6.5)c are not unique and also do not effect $\underset{\rightarrow}{\boldsymbol{y}}^{(1)}$, $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}$ (but do effect $\underset{\rightarrow}{\boldsymbol{y}}^{(2)}$, $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(2)}$), and hence the question arises as to whether the freedom in the choice of M and N can be used to advantage. The answer to this is contained in constraints (6.34), according to which the future control moves are seen to be restricted by the vectors $\boldsymbol{R}^{\#}$ and $\boldsymbol{U}^{\#}$. To optimize performance, one needs to maximize the intervals of (6.34); this implies that one needs to make the elements of $\boldsymbol{R}^{\#}$ and $\boldsymbol{U}^{\#}$ as small as possible. In this section, we explore this issue and suggest algorithms for the optimal choice of M and N. The reader is reminded that although the solutions $M(z)$, $N(z)$ of eqn. (6.5)c and the solutions $M^{\#}(z)$, $N^{\#}(z)$ of eqn. (2.4) belong to the same family (they satisfy the same identity), as explained earlier, they can be chosen to be different; thus the degrees of freedom in $M(z)$, $N(z)$ and those in $M^{\#}(z)$, $N^{\#}(z)$ can be deployed for different purposes.

6.3.3.1.1 The degrees of freedom and the cost. The totality of solutions to eqn. (6.5)c is given as $N(z)=X(z)+\alpha(z)Q(z)$, and $M(z)=Y(z)-z^{-1}b(z)Q(z)$, where Q denotes a

polynomial, and X and Y are the minimal order solutions for N and M. Thus we have:

$$N_i=(\alpha_i Q_0+\alpha_{i-1}Q_1+\ldots+\alpha_0 Q_i)+X_i; \qquad M_i=-(b_{i-1}Q_0+b_{i-2}Q_1+\ldots+b_0 Q_{i-1})+Y_i \tag{6.50}$$

where the subscript i is used to denote coefficients of the i^{th} power of z^{-1}. Clearly, the Q_i's denote the available degrees of freedom which can be used to minimize the elements of $\boldsymbol{R}^{\#}$ and $\boldsymbol{U}^{\#}$. Now, from eqns. (6.43),(6.44), it should be apparent that the optimal choice of Q is different for the minimization $\boldsymbol{R}^{\#}$ than it is for $\boldsymbol{U}^{\#}$, and indeed is different for the minimization of the different elements of $\boldsymbol{R}^{\#}$ or $\boldsymbol{U}^{\#}$. Thus, the general optimization problem can be posed as:

$$\min_{Q} J_Q \; ; \qquad J_Q=\left\| \begin{bmatrix} W_R \boldsymbol{R}^{\#} \\ W_U \boldsymbol{U}^{\#} \end{bmatrix} \right\|_{\infty} \tag{6.51}$$

where W_R, W_U are two diagonal matrices of weights intended to adjust the emphasis between the different elements of $\boldsymbol{R}^{\#}$ and $\boldsymbol{U}^{\#}$. For example, for a problem with stringent rate constraints, W_R could be chosen to be larger than W_U. For most practical applications, given the monotonically increasing nature of the elements of both $\boldsymbol{R}^{\#}$ and $\boldsymbol{U}^{\#}$, a simple and sensible strategy would be to place a penalty on the largest (the last) element only of $\boldsymbol{R}^{\#}$ and $\boldsymbol{U}^{\#}$ for which the cost assumes the form:

ζ_t unknown: $$J_Q'=\left\| \left[w_R(|N_0|+\ldots+|N_k|),\ w_U(|N_0|+\ldots+|N_0+\ldots+N_k|)\right] \right\|_{\infty} \tag{6.52}$$

ζ_t known: $$J_Q'=\left\| \left[w_R(|N_0|+|N_1-N_0|+\ldots+|-N_k|),\ w_U(|N_0|+\ldots+|N_k|)\right] \right\|_{\infty} \tag{6.53}$$

Without loss of generality, we present the minimization of J_Q' for ζ_t known only.

6.3.3.1.2 Rate constraints or absolute constraints only. Consider rate or absolute limits only, say rate limits only. Then w_R in eqn. (6.53) can be taken to be 1 and w_U=0; and so, using eqn. (6.50)a, J_Q' can be written as the 1-norm of a linear

functional of $\boldsymbol{v}_Q$, the vector of coefficients of Q:

$$J_Q' = \| F\boldsymbol{v}_Q + \boldsymbol{h} \|_1; \quad \boldsymbol{F}_i^T = [\alpha_{i-1} - \alpha_{i-2}, \alpha_{i-1} - \alpha_{i-2}, \ldots, \alpha_0, 0, \ldots, 0]; \quad h_i = X_{i-1} - X_{i-2} \tag{6.54}$$

where $\boldsymbol{F}_i^T$ denotes the i^{th} row vector of F; no coefficient can have a negative index, and all coefficients with indices which exceed the order of $\alpha(z)$ or $X(z)$ are zero. This problem can be solved using linear programming (see, for example [11]); here, we develop briefly a particular implementation of the procedure for the minimization of J_Q'. First, note that, since the degree of $N(z)$ is always greater than the degree of $Q(z)$, F will have more rows than columns, so we can write:

$$F\boldsymbol{v}_Q + \boldsymbol{h} = \boldsymbol{w} \;\Leftrightarrow\; \begin{matrix} F_1\boldsymbol{v}_Q + \boldsymbol{h}_1 = \boldsymbol{w}_1 \\ F_2\boldsymbol{v}_Q + \boldsymbol{h}_2 = \boldsymbol{w}_2 \end{matrix} \;\Leftrightarrow\; \begin{matrix} \boldsymbol{v}_Q = F_1^{-1}(\boldsymbol{w}_1 - \boldsymbol{h}_1) \\ A'\boldsymbol{w} = \boldsymbol{b}'; \end{matrix} \qquad A' = [F_2F_1^{-1} \quad -I], \; \boldsymbol{b}' = A'\boldsymbol{h} \tag{6.55}$$

where the subscripts 1 and 2 indicate a partition of eqn. (6.55)a into the first n_Q+1 equations and the remainder, with n_Q+1 being the dimension of $\boldsymbol{v}_Q$; F_1 is a square lower triangular matrix with 1's along its diagonal and is invertible. Now, define a non-negative vector $\boldsymbol{z}$ (as shown below) which is twice as long as $\boldsymbol{w}$ and is such that the top half equals $\boldsymbol{w}$ with all negative elements replaced by zero and the bottom half equals $-\boldsymbol{w}$ with all (originally) positive elements replaced by zero. Then solve $A'\boldsymbol{w} = \boldsymbol{b}'$ to get:

$$\boldsymbol{w} = P\boldsymbol{z}; \qquad P = [I_{k+2} \quad -I_{k+2}]; \qquad \boldsymbol{z} = \boldsymbol{z}_p + K\boldsymbol{v} \geq \boldsymbol{0} \tag{6.56}$$

where $\boldsymbol{z}_P$ denotes a particular solution of $A'P\boldsymbol{z} = \boldsymbol{b}'$ ($\boldsymbol{z}_p$ could be taken to be the vector of positive and negative parts of $\boldsymbol{h}$, s.t. $P\boldsymbol{z}_p = \boldsymbol{h}$), and K is a matrix representation of the kernel of $A'P$.

Theorem 6.8: The optimal solution, $\boldsymbol{v}^*$, of the linear programming problem:

$$\min_{\boldsymbol{v}} \boldsymbol{f}^T\boldsymbol{v}, \quad s.t. \; -K\boldsymbol{v} \leq \boldsymbol{z}_p \qquad \text{where} \quad \boldsymbol{f}^T = \boldsymbol{1}^T K \tag{6.57}$$

yields the vector $\boldsymbol{v}_Q^*$ which minimizes the cost J_Q' of eqn. (6.53) for $w_R = 1$, $w_U = 0$ as:

$$v_Q^* = F_1^{-1}(w_1^* - h_1); \qquad w_1^* = P_1 z^*; \qquad z^* = z_p + K v^* \tag{6.58}$$

Proof: Since z comprises the positive and minus the negative parts of the elements of $w = Fv_Q + h$, the sum of the absolute value of the elements of w can be written as the sum of the elements of z:

$$J_Q' = \|w\|_1 = 1^T z = 1^T z_p + 1^T K v = 1^T z_p + f^T v \tag{6.59}$$

The minimization of J_Q' requires the minimization of $f^T v$ ($1^T z_p$ is constant), as per the cost part of (6.57). By definition, z must be non-negative (eqn. (6.56)c); this is the constraint part of (6.57). □

The manner of solution presented in Theorem 6.8 is chosen for clarity; modifications to this can lead to more efficient implementations, but Theorem 6.8 is easy to implement and involves a small computational burden, especially for low order $Q(z)$. We also note that the treatment of the case of input absolute constraints *only* is similar. Finally we remark that in some cases there are well defined limits on the optimal value of J_Q'; the result below, considers two cases.

Corollary 6.6 For $a(z)$ stable, the optimal value of the cost of (6.53) over all stable Q is $1/|b(1)|$ for $w_R=0$, $w_U=1$, and 0 for $w_R=1$, $w_U=0$. In general, these values are not attainable for $a(z)$ unstable.

Proof: We consider the two cases separately:

Case 1: $w_R=0$, $w_U=1$. From the definitions of $\alpha(z)$, b(z), and bezout identity (6.5)c for $z=1$, we have $N(1)=1/b(1)$. Combining this with the triangle inequality we deduce that:

$$1/|b(1)| = |N(1)| = |\sum_i N_i| \le \sum_i |N_i| = \|v_N\|_1 \qquad \textbf{(6.60)}$$

Thus, $|1/b(1)|$ is a lower bound on the cost of eqn. (6.53) for $w_R=0$, $w_U=1$. Next we show that for $a(z)$ stable this bound can be attained, ie. that there exists a stable Q for which N_i share the same sign for all i. Let $q(z)=a(z)Q(z)$, assume that $b(1)>0$, and consider the conditions implied by $N_i \ge 0$; these, given that $N(z)=X(z)+\alpha(z)Q(z)=X(z)+\Delta(z)q(z)$, can be written as:

$$-X_0 \le q_0; \quad -X_1+q_0 \le q_1; \quad \ldots \quad -X_k+q_{k-1} \le q_k; \quad q_{k+i} \le q_{k+i+1} \quad \text{for} \quad i=0,1,\ldots \qquad \textbf{(6.61)}$$

The implied $Q(z)=q(z)/a(z)$ must be stable, and so q_j must tend to zero as j tends to infinity. This, together with the last inequality of (6.61), imply $q_{k+i}=0$ for $i=1,2,\ldots$, hence (6.61) reduces to:

$$-X_0 \le q_0; \quad -X_1+q_0 \le q_1; \quad \cdots \quad -X_{k-2}+q_{k-3} \le q_{k-2}; \quad -X_{k-1}+q_{k-2} \le q_{k-1} \le X_k \qquad q_{k+i}=0 \text{ for } i=0,1,\ldots \qquad \textbf{(6.62)}$$

On account of the assumption that $b(1)>0$, this set of conditions admits at least one solution, given by $q_{i-1}=X_{i-1}+q_{i-2}$ for $i=0,1,\ldots,k$ and $q_{k+i}=0$ for $i=0,1,\ldots$ The same proof holds for $b(1)<0$, only now, the direction of all the inequalities must be reversed.

The same arguments apply when $a(z)$ is unstable, but now $q(z)$ must cancel the unstable poles of $a(z)$, thus implying some equality constraints which, in general, will not be consistent with (6.62).

Case 2: $w_R=1$, $w_U=0$. As with the previous case, we have that $N_i=X_i+q_i-q_{i-1}$, and clearly the first k of these can be set equal to ϵ; the last coefficient will be constrained by the condition $N(1)=1/b(1)$ and will be given as $N_{k+1}=1/b(1)-k\epsilon$. Thus for $\epsilon=1/[(k+1)b(1)]$, all the coefficients of $N(z)$ will be equal to ϵ, so that J_Q' will reduce to $J_Q' = |N_0| + |N_{k+1}| = 2\epsilon = 2/[(k+1)b(1)]$. This value of the cost can be made to be arbitrarily small as k becomes arbitrarily large. Such arguments cannot be used for the case of $a(z)$ unstable,

because once again, the coefficients of $q(z)$ will have to satisfy an additional set of equality constraints that ensure the cancellation of the unstable poles of $a(z)$. □

6.3.3.1.3 Rate and absolute constraints. In the presence of both types of constraints, the cost J_Q' (for ζ_t known) may be written as:

$$J_Q' = \left\| \begin{bmatrix} \|Gv_Q+\gamma\|_1 \\ \|Hv_Q+\delta\|_1 \end{bmatrix} \right\|_\infty ; \quad \begin{bmatrix} G_i^T = w_R[\alpha_{i-1}-\alpha_{i-2},\ldots,\alpha_0,0,\ldots,0] \\ H_i^T = w_U[\alpha_{i-1},\alpha_{i-2},\ldots,\alpha_0,0,\ldots,0] \end{bmatrix}; \quad \begin{bmatrix} \gamma_i = w_R(X_{i-1}-X_{i-2}) \\ \delta_i = w_U X_i \end{bmatrix} \tag{6.63}$$

Next, by defining

$$F=[G^T \;\; H^T]^T; \qquad h=[\gamma^T \;\; \delta^T]^T \tag{6.64}$$

we can reiterate eqns. (6.55) and (6.56) and state the following result.

Theorem 6.9: The optimal solution, $[v^{*T}, \rho^*]^T$, of the linear programming problem:

$$\min_{v,\rho} f^T \begin{bmatrix} v \\ \rho \end{bmatrix}, \quad s.t. \quad \begin{bmatrix} P'K & -1 \\ -P'K & -1 \\ -K & 0 \end{bmatrix} \begin{bmatrix} v \\ \rho \end{bmatrix} \le \begin{bmatrix} -P'z_p \\ P'z_p \\ z_p \end{bmatrix}; \tag{6.65}$$

$$f^T=[0 \;\; \cdots \;\; 0 \;\; 1], \qquad P' = \begin{bmatrix} 1\cdots1,0\cdots0,1\cdots1,0\cdots0 \\ 0\cdots0,1\cdots1,0\cdots0,1\cdots1 \end{bmatrix}$$

yields the vector v_Q^* which minimizes the cost J_Q' of eqn. (6.53) as:

$$v_Q^* = F_1^{-1}(w_1^* - h_1); \qquad w_1^* = P_1 z^*; \qquad z^* = z_p + Kv^* \tag{6.66}$$

Proof: With the definitions above, it is easy to show that the cost J_Q' is given as:

$$J_Q' = \| P'z \|_\infty = \| P'z_p + P'Kv \|_\infty \tag{6.67}$$

If the value of this ∞-norm were ρ, then both elements of $P'(z_p+Kv)$ should lie in $[-\rho,\rho]$:

$$-\rho\mathbf{1} \le P'z_p + P'Kv \le \rho\mathbf{1} \tag{6.68}$$

This, together with eqn. (6.56)c, leads to the constraint part of (6.65). Our aim is to minimize the J_Q' of eqn. (6.67), which has been assumed to be equal to ρ; the cost part

of (6.65), for the particular choice of $\boldsymbol{f}^T$, calls for the minimization of this value ρ. □

Theorems 6.8, 6.9 address the problem of minimizing the cost of eqn. (6.53), but can be extended easily to handle the general cost of eqn. (6.51); the procedure is exactly the same, but the expressions for the various matrices and vectors become more complicated.

Example 6.3 Consider a model for which the numerator and denominator polynomials are given as:

$$a(z)=1-2.2z^{-1}+0.09z^{-2}+0.252z^{-3}; \quad b(z)=2+0.45z^{-1}+z^{-2} \tag{6.69}$$

and let $n_c=3$, $R=0.05$, $U_o=0$, $U=0.3$, $d=0.007$, $n_Q=1$, $w_R=2$, $w_U=1$ (for ζ_t unknown) and $w_R=1$, $w_U=2$ (for ζ_t known). Then the minimal order solution of the bezout identity of eqn. (6.5)c, $X(z)$, and the optimal choice for $Q(z)$ (as per Theorem 6.9) are:

$$X(z)=1.4409-1.2191z^{-1}-0.0625z^{-2}+0.1306z^{-3}; \quad Q_{opt}(z)=-0.4986-0.3763z^{-1} \tag{6.70}$$

For this example and the particular choice of weights, w_R and w_U, the optimal solution for Q is the same for both ζ_t known and unknown. The benefit derived from this optimal choice of $Q(z)$ is illustrated in Figure 6.6 (for ζ_t unknown) and Figure 6.7 (for ζ_t known) which depict the actual constraints, as defined by R, U_o and U (solid lines), and the *a priori* constraints, for $Q(z)=0$ (dash-dotted lines) and $Q_{opt}(z)$ (dashed lines); in both figures, the left plot concerns rate limits, whereas the right plot deals with absolute input constraints. For increased detail, the right plots show the upper bounds only; the lower bounds are symmetrical about $U_0=0$. Clearly, $Q_{opt}(z)$ has increased the interval within which the vector of future control moves must lie, thus allowing MCSGPC more freedom to improve performance; this is illustrated in the design study of Section 6.3.4.

6.3.3.2 Deploying the *a priori* degrees of freedom

MCSGPC, as per Algorithm 6.1, constrains the current value of c, c_t (the first element of $\underset{\rightarrow}{c}$), **as well as all the future values of** c, c_{t+i} for $i=1,2,\ldots,n_c$-1 (all remaining elements of $\underset{\rightarrow}{c}$), according to the *a priori* conditions (and hence the *a posteriori* conditions also) of Theorem 6.6. This is convenient in that it keeps the complexity of the relevant optimization problem to a minimum. However, c_{t+i} can actually be changed at the next sampling instant, ie. at $t+1$, and this extra freedom can be deployed to lessen the feasibility burden on c_{t+i} at the current time, t. Of course this deployment refers to $i>0$ only, because c_t will be chosen and implemented now; no further corrective action will be available in the future with respect to this value of c. These ideas can be made concrete with reference to eqn. (6.42) of Lemma 6.4 according to which we may write:

$$\begin{bmatrix} \underset{\rightarrow}{c}(t+1) \\ c_\infty(t+1) \end{bmatrix} = \begin{bmatrix} \underset{\rightarrow}{c}(t+1\,|\,t) \\ c_\infty(t) \end{bmatrix} + \boldsymbol{x} \qquad \textbf{(6.71)}$$

where the definitions of $c(.)$ are analogous to those given in Definition 6.1; in particular the leading elements of $\underset{\rightarrow}{c}(t+1\,|\,t)$ are given by the 2nd, 3rd,...,n_c^{th} elements of $\underset{\rightarrow}{c}(t)$ and the last element is $c_\infty(t)$. The argument made is that the vectors $\underset{\rightarrow}{\Delta u}^{(1)}(t+1)$ and $\underset{\rightarrow}{u}^{(1)}(t+1)$ of eqn. (6.42) depend on $\boldsymbol{x}$, and so the extra freedom available in $\boldsymbol{x}$ can be used to relax the feasibility constraints imposed on $\underset{\rightarrow}{c}(t)$. In this context, the feasibility requirement becomes that: (i) $\underset{\rightarrow}{\Delta u}^{(1)}(t)$, $\underset{\rightarrow}{u}^{(1)}(t)$ lie between actual constraints (2.28); (ii) $\underset{\rightarrow}{\Delta u}(t+1)$, $\underset{\rightarrow}{u}(t+1)$ also lie between the actual constraints; (iii) requirement (ii) can be satisfied at the next time instant (and hence at all subsequent times). Requirement (ii) implies that $\underset{\rightarrow}{\Delta u}^{(1)}(t+1)$ and $\underset{\rightarrow}{u}^{(1)}(t+1)$ should be subject to the appropriate *a posteriori* feasibility conditions; requirement (iii), in conjunction with (ii), implies that $\underset{\rightarrow}{\Delta u}^{(1)}(t+1)$ and $\underset{\rightarrow}{u}^{(1)}(t+1)$

should be subject to the appropriate *a priori* conditions; whereas condition (i) has the effect of replacing the bordered inequality (6.34) by the wider constraints of inequality (2.28). In other words, the burden of future feasibility is, in effect, passed from c_{t+i}, $i>0$ onto $\boldsymbol{x}$ through $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}(t+1)$ and $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1)$. On the surface, it appears that $\underset{\rightarrow}{\boldsymbol{c}}(t)$ is only subject to the constraints implied by the actual limits and therefore takes no account of disturbances. This is exactly what one should expect with respect to c_{t+i} $i>0$, because by eqn. (6.71), $\boldsymbol{x}$ can alter these values of c; but one may feel concern with respect to c_t, because $\boldsymbol{x}$ cannot alter its effects. However, $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}(t+1\,|\,t)$, $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1\,|\,t)$ depend on c_t, and thus the *a priori* constraints imposed on $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}(t+1)$, $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1)$ constrain, indirectly, the values that c_t is allowed to assume. Thus the first appearance, that $\underset{\rightarrow}{\boldsymbol{c}}(t)$ is subject to the actual constraints only, is not correct; indeed what is happening is that $\underset{\rightarrow}{\boldsymbol{c}}(t)$ is subject to constraints which are tighter than those implied by the actual limits, but looser than those implied by the bordered constraints of Theorem 6.6. One can therefore expect improved performance; the price to be paid for this is increased computational complexity: MWLS now involves $\boldsymbol{x}$ as well as $\underset{\rightarrow}{\boldsymbol{c}}(t)$.

It was stated that, for $\underset{\rightarrow}{\Delta \boldsymbol{u}}(t+1)$, $\underset{\rightarrow}{\boldsymbol{u}}(t+1)$ to lie within the actual constraints and for this to be possible at the next time instant, $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}(t+1)$ and $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1)$ should be subject to the appropriate *a priori* conditions. The result below implements this new requirement.

Lemma 6.9 At time t, the guarantee of i) STF at $t+1$, ii) its maintenance at all subsequent time instants, and iii) feasibility of $\underset{\rightarrow}{\Delta \boldsymbol{u}}(t+1)$, $\underset{\rightarrow}{\boldsymbol{u}}(t+1)$ is ensured if a vector $\boldsymbol{x}$ can be found such that:

$$\begin{aligned} -\boldsymbol{Rq}+\boldsymbol{R}^{\#} &\le \underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}(t+1\,|\,t)+\left[\Gamma_{\alpha} \quad \boldsymbol{\theta}_{\alpha}\right]\boldsymbol{x} \qquad \le \boldsymbol{Rq}-\boldsymbol{R}^{\#} \\ -\boldsymbol{Uq}+\boldsymbol{U}^{\#} &\le \underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1\,|\,t)+\left[\Gamma_{a} \quad \boldsymbol{\theta}_{a}\right]\boldsymbol{x}-\boldsymbol{U}_0\boldsymbol{q} \le \boldsymbol{Uq}-\boldsymbol{U}^{\#} \end{aligned} \tag{6.72}$$

where:

$$\begin{array}{lll} \zeta_t \text{ unknown:} & R_1^{\#}=2d|N_0|, & R_i^{\#}=R_{i-1}^{\#}+2d|N_{i-1}|; \\ & U_1^{\#}=2d|N_0|, & U_i^{\#}=U_{i-1}^{\#}+2d|N_0+\cdots+N_{i-1}| \\ \zeta_t \text{ known:} & R_1^{\#}=d|N_0|, & R_i^{\#}=R_{i-1}^{\#}+d|N_{i-1}-N_{i-2}|; \\ & U_1^{\#}=d|N_0|, & U_i^{\#}=U_{i-1}^{\#}+d|N_{i-1}| \end{array} \qquad i>1 \tag{6.73}$$

Furthermore, eqns. (6.73) define the smallest borders for which STF at $t+1$, $t+2$,... can be guaranteed.

Proof: This parallels that given for Theorem 6.6, except that now eqns. (6.7) and (6.8) must run one step forward in time, and the vectors $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}$ and $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}$ must be replaced by $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}(t+1)$ and $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1)$ (eqn. (6.42)). Clearly, the terms $\boldsymbol{f}_{\Delta u}$ and $\boldsymbol{f}_{u}$ are unknown and must be removed from $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(1)}(t+1)$ and $\underset{\rightarrow}{\boldsymbol{u}}^{(1)}(t+1)$ and included in $\underset{\rightarrow}{\Delta \boldsymbol{u}}^{(2)}$ and $\underset{\rightarrow}{\boldsymbol{u}}^{(2)}$ (and thus included in the borders). The remainder of the proof is identical to that of Theorem 6.6. □

Algorithm 6.2 (MCSGPC with disturbances)

This is identical to Algorithm 4.4 with the added constraint of (6.72),(6.73).

As noted above, this algorithm removes the borders of Algorithm 6.1 (thus returning to the original MCSGPC constraints), but indirectly inserts an additional constraint on c_t via (6.72).

Theorem 6.10 Algorithm 6.2 is guaranteed to remain stable in the presence of the disturbances of eqn. (6.1) and condition (6.2). Furthermore, if the disturbance, ζ_t, reaches a steady-state, then Algorithm 6.2 has guaranteed asymptotic tracking.

Proof: This runs along lines similar to the arguments given for Theorem 6.7 (Lemmata 6.6 and 6.7), except that now, STF and disturbance handling are ensured by Lemma 6.8, instead of Theorem 6.6. □

6.3.4 Design study

In this section, we describe a design study which illustrates the stability properties of Algorithms 6.1 and 6.2 as well as demonstrates how optimizing $Q(z)$ and deploying the vector $\boldsymbol{x}$ (of Algorithm 6.2) leads to better performance. The $a(z)$, $b(z)$ polynomials, as well as the constraint limits to be used, are the same as those given in Example 6.3. The choices for the parameters n_c and λ are 3 and 1 respectively, whereas the disturbance, ξ_t, consists of a negative impulse of 0.007 applied at $t=4$ followed by a positive impulse of 0.014 applied at $t=5$ and results in the ζ_t signal shown in the left plots of Figure 6.8-Figure 6.12 (scaled by a factor of 20). The simulation results for MCSGPC are shown in Figure 6.8 and can be seen to be unstable. In this figure and throughout this study, the left plot will depict output responses (solid lines), set-point trajectories (dashed lines) and disturbance signals (dash-dotted lines, scaled by 20), whereas the right plot will depict the input trajectories (solid lines) and the trajectory of the control increments (dash-dotted lines, scaled by a factor of 6). The instability of Figure 6.8 is easy to explain: at the time when the disturbance is applied, MCSGPC is driving hard against the rate limits and so no control action is available for handling the disturbance. As a consequence, the problem becomes STIF, and this in turn results in instability.

In contrast to this, Algorithm 6.1 reserves just enough control action so as to handle the worst case disturbance sequences and hence can cope well with the situation described above. This is illustrated in Figure 6.9 (for ζ_t unknown) and Figure 6.10 (for

ζ_t known); the responses of both figures are good, but those of Figure 6.10 are of course better, since the algorithm makes use of knowledge of ζ_t.

These plots are for $Q(z)=0$, and further improvements in the response can be brought about through the use of the optimal $Q(z)$ of eqn. (6.70)b; these simulation results are shown in Figure 6.11 and Figure 6.12 and can be seen to be better (faster) than those presented in Figure 6.9 and Figure 6.10. Further evidence of this improvement is presented in Table 6.1, which records the values assumed by the cost:

$$J_{total}=\sum_{t=1}^{runtime}(r_{t+1}-y_{t+1})^2+\lambda(\Delta u_t)^2 \qquad \textbf{(6.74)}$$

where r_t denotes the set-point sequence defined by $r_i=0$ for $i=1,2,\ldots,7$, and $r_i=1$ for $i>7$. Table 6.1 also compares the performance of Algorithm 6.2, for which the results are similar to those of Figure 6.9-Figure 6.12 and, therefore, are not shown.

		ζ_t unknown	ζ_t known
Algorithm 6.1	$Q=0$	1.0084	0.8002
	$Q=Q_{opt}$	0.7100	0.7279
Algorithm 6.2	$Q=0$	0.9857	0.7744
	$Q=Q_{opt}$	0.6768	0.6554

Table 6.1: Comparison of costs using Algorithms 6.1 and 6.2 with $Q=0$ and $Q=Q_{opt}$

Clearly, utilizing the degrees of freedom available in Q brings about a very significant reduction (about 30% for ζ_t unknown and about 9% for ζ_t known) in the cost of eqn. (6.74); the deployment of the degrees of freedom in the vector $\boldsymbol{x}$ of Algorithm 6.2 brings about an additional, but smaller, improvement (about 5% for ζ_t unknown and about 10% for ζ_t known).

In conclusion, therefore, both Algorithms 6.1 and 6.2 produce results which

illustrate the stability and asymptotic tracking properties of Theorems 6.7 and 6.10. Furthermore, the algorithms produce some very good responses which are further improved through the use of Q_{opt}. Finally, Algorithm 6.1, though marginally outperformed by Algorithm 6.2, is numerically simple and, therefore, is the more attractive alternative.

6.4 Chapter summary

Disturbances are omnipresent and, for systems subject to constraints, must not be neglected. In this chapter, we have defined the inputs which are necessary to reject future bounded disturbances, and then used this knowledge to: i) develop explicit *a priori* stability conditions for systems with just one unstable pole, and ii) derive and deploy borders in MCSGPC which reserve enough control authority to reject these disturbances and yet release enough incremental control authority to ensure that the effects of the current disturbance do not make the problem infeasible at the following time instant.

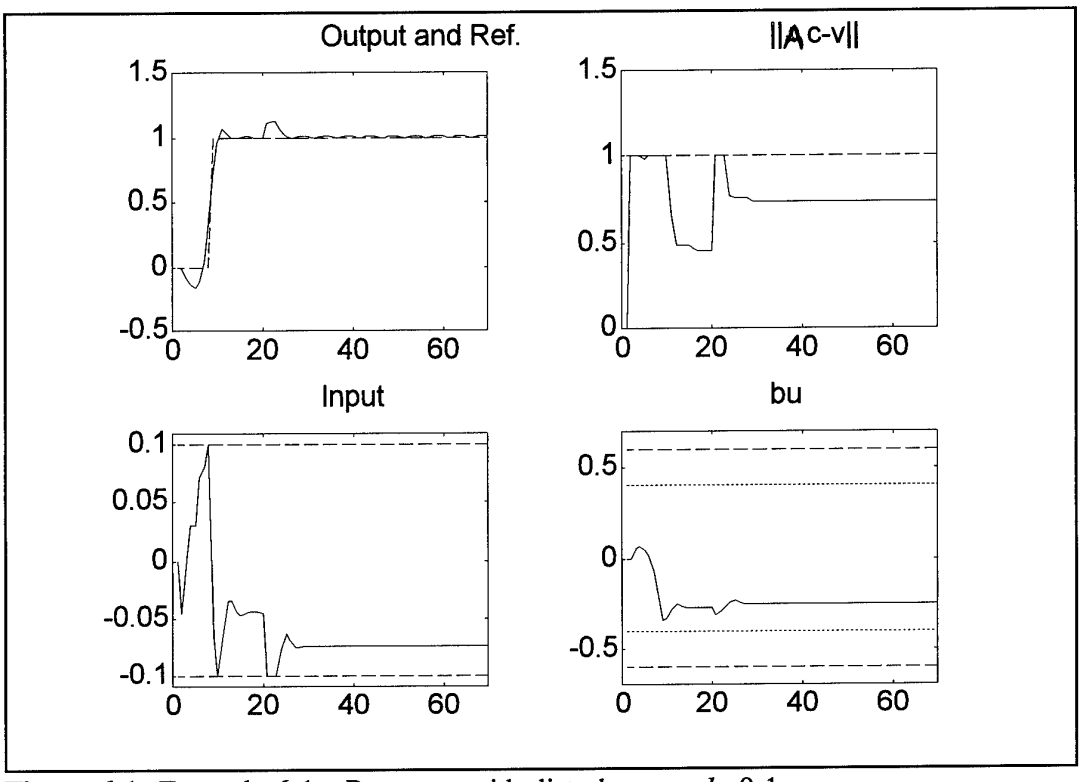


Figure 6.1 Example 6.1 - Response with disturbance - d=0.1

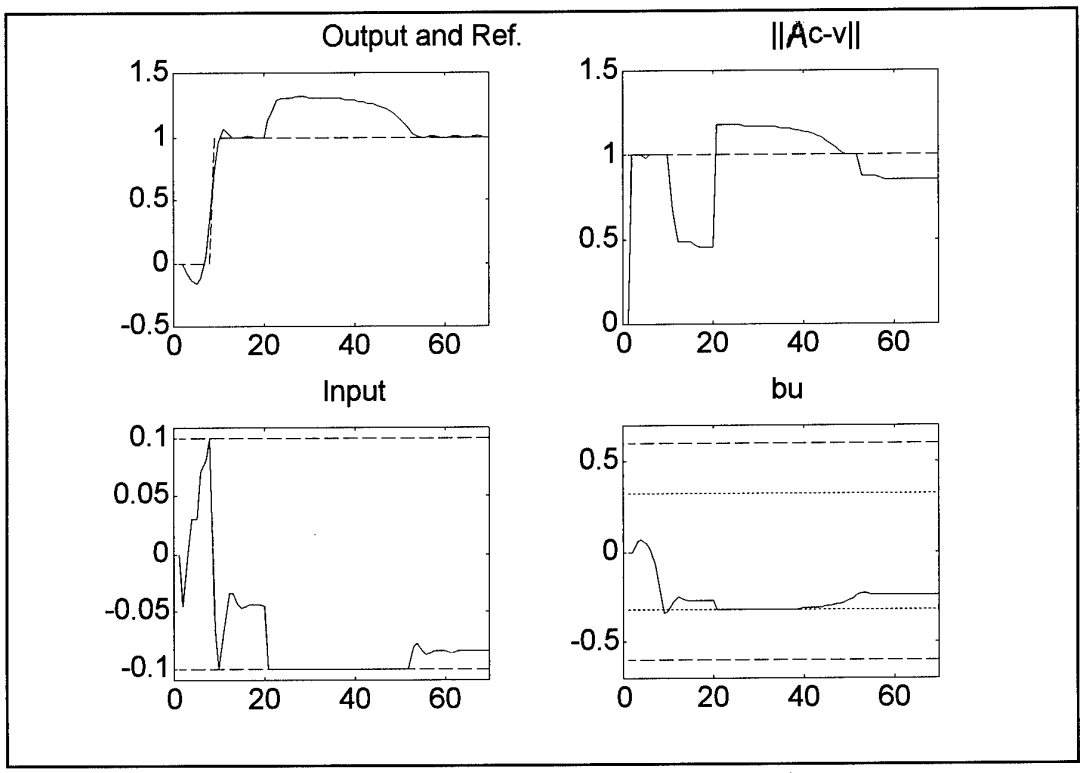


Figure 6.2 Example 6.1 - Response with disturbance - d=0.14

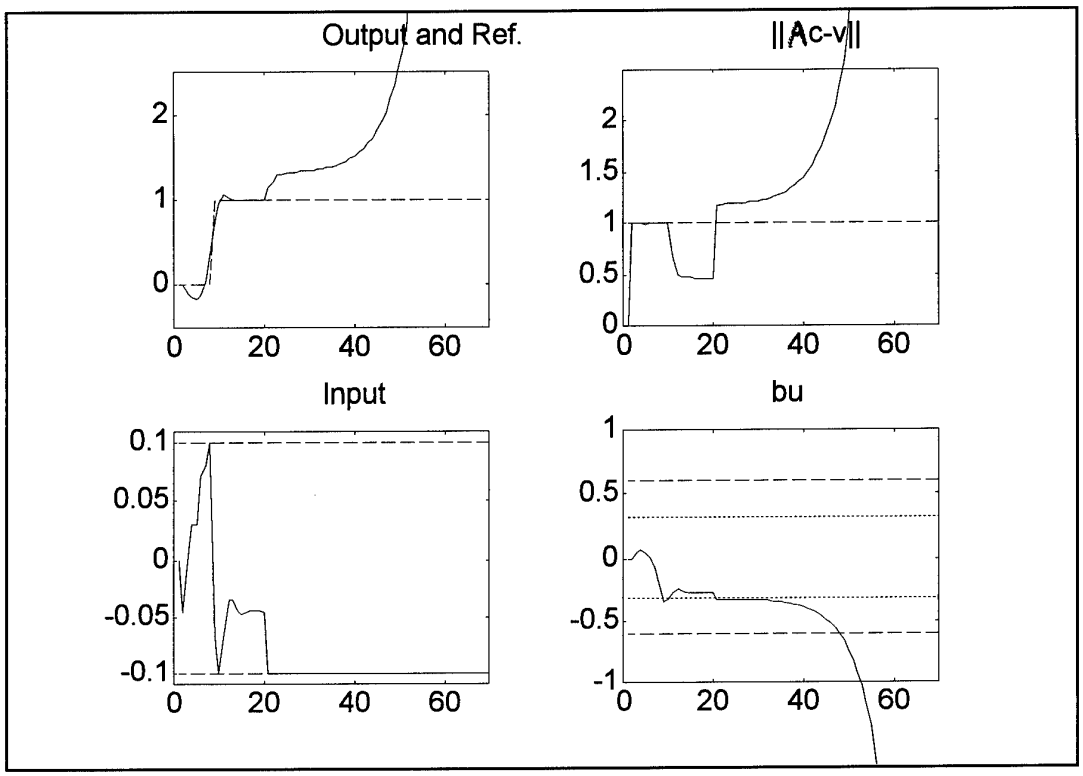


Figure 6.3 Example 6.1 - Response with disturbance - d=0.141

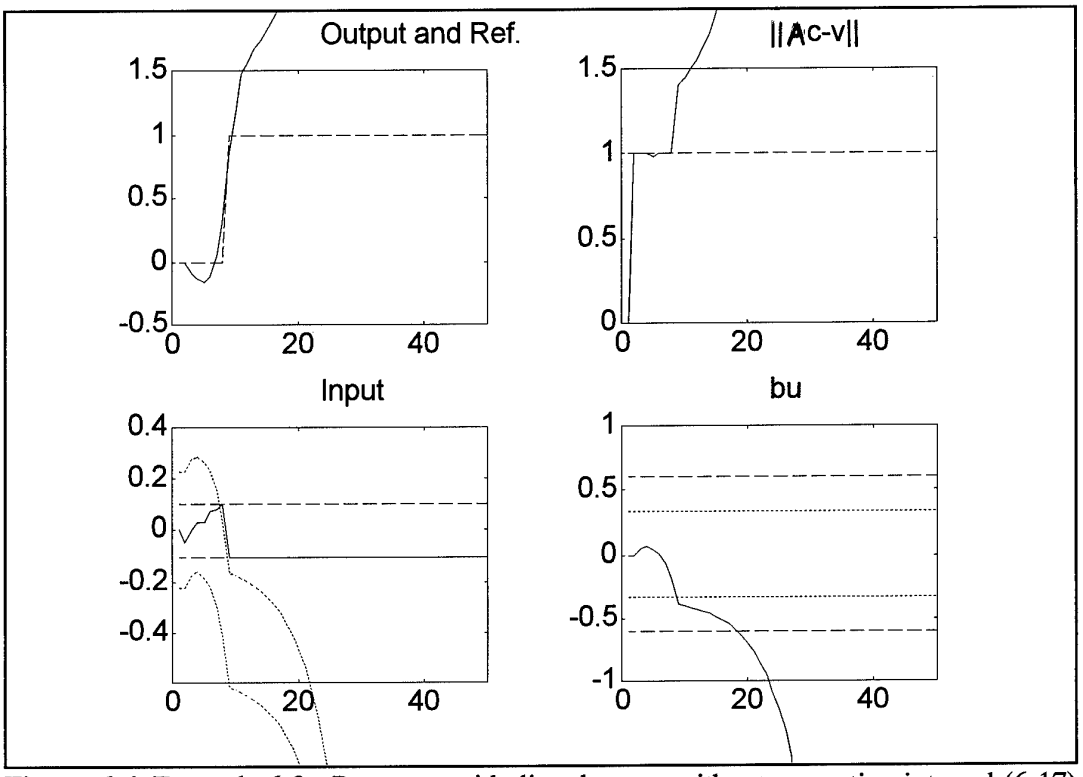


Figure 6.4 Example 6.2 - Response with disturbance - without respecting interval (6.17)

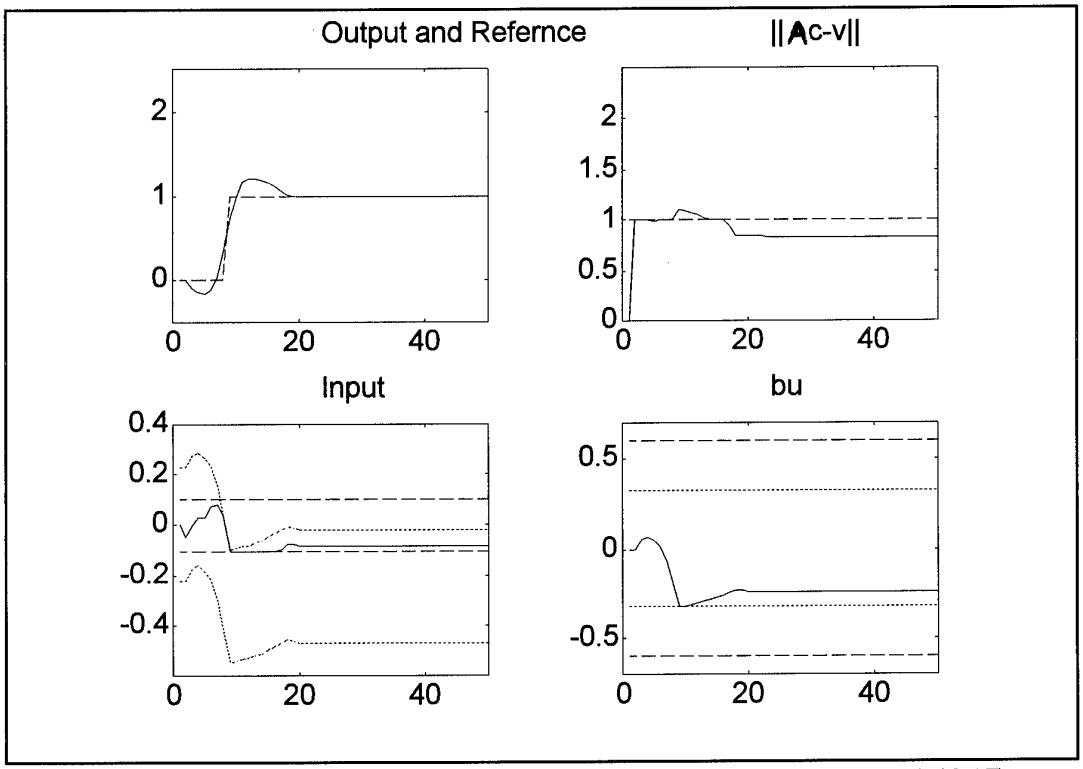


Figure 6.5 Example 6.2 - Response with disturbance - respecting interval (6.17)

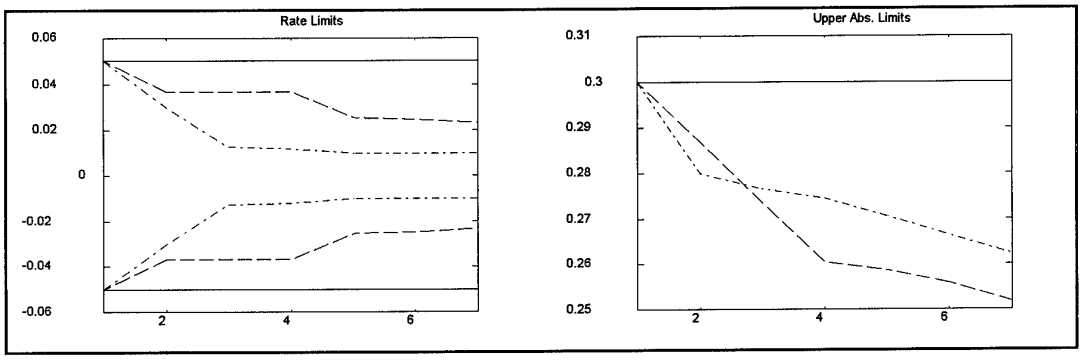


Figure 6.6 Example 6.3 - Borders for ζ_t unknown

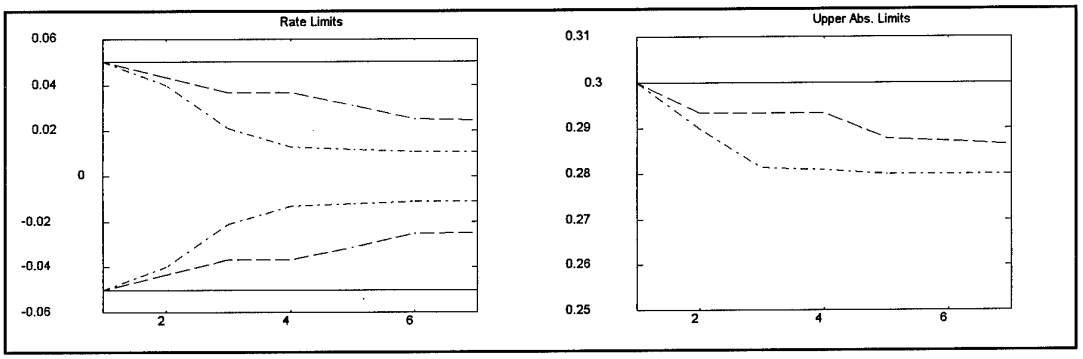


Figure 6.7 Example 6.3 - Borders for ζ_t known

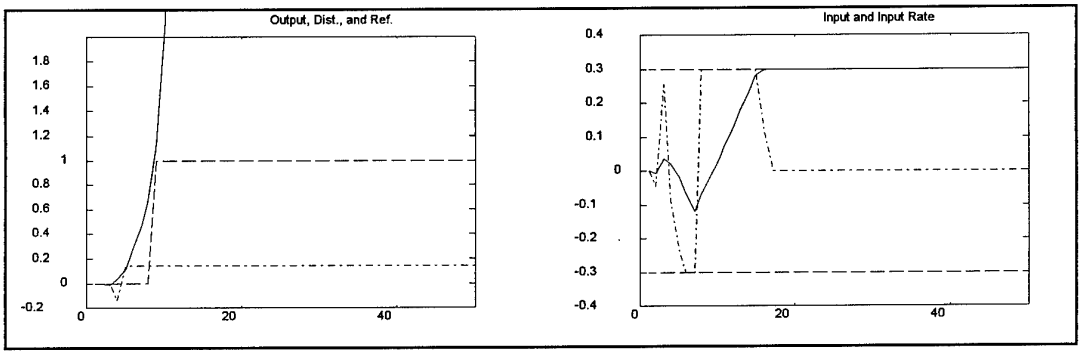


Figure 6.8 Design Study - MCSGPC with no borders

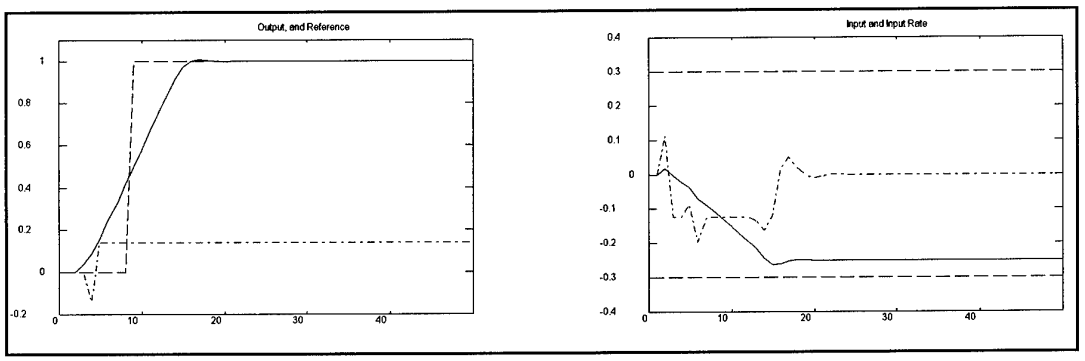


Figure 6.9 Design Study - Bordered MCSGPC (ζ_t unknown, Q=0)

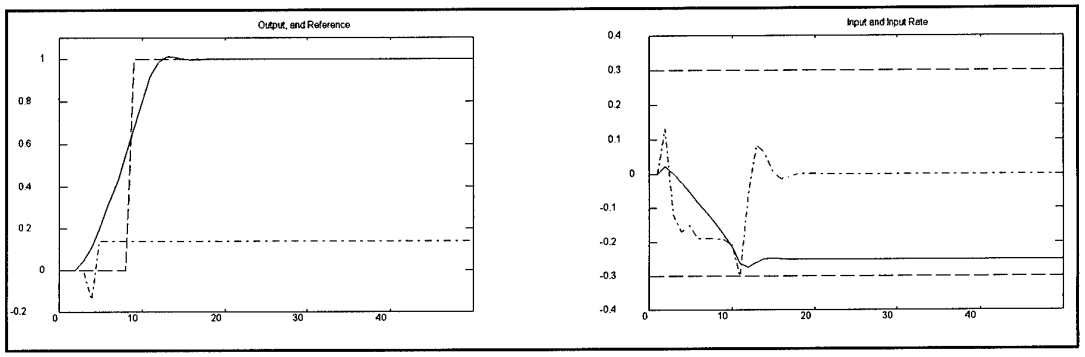


Figure 6.10 Design Study - Bordered MCSGPC (ζ_t known, Q=0)

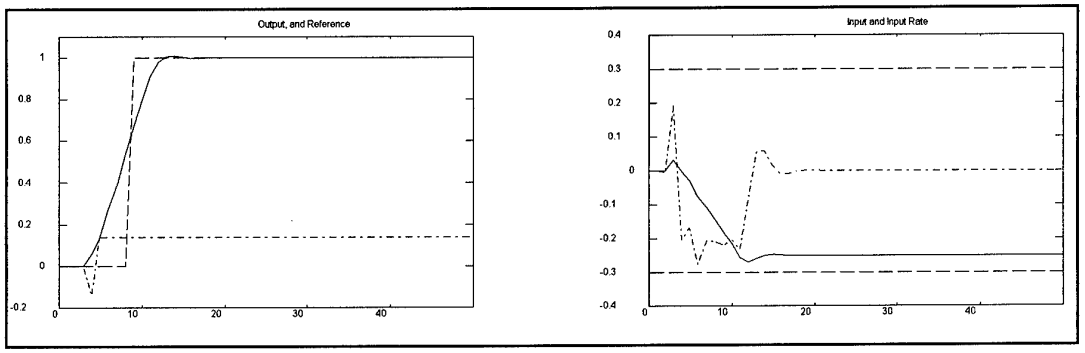


Figure 6.11 Design Study - Bordered MCSGPC (ζ_t unknown, Q_{opt})

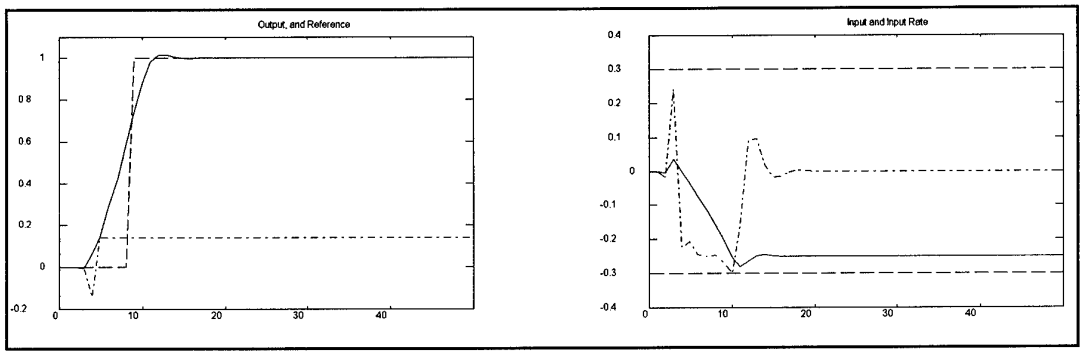


Figure 6.12 Design Study - Bordered MCSGPC (ζ_t known, Q_{opt})

Chapter 7

Conclusion

Here, we summarize the important results of this thesis and then briefly discuss some of the remaining open problems.

7.1 Thesis summary

The problem of feasibility was investigated in Chapter 3, and necessary and sufficient conditions for stability were developed. We derived *a priori* stability conditions which, for systems with only one unstable pole, provided explicit conditions for retaining long term feasibility (LTF). Obviously, *any* control strategy which ignores these conditions and violates them will be unstable. Later (in Chapter 6), we also showed that these results carry over to the more general case of systems which are subject to disturbances and showed how they could be used to avoid instability. We then (in Chapter 3) developed general stability conditions for systems with any number of unstable poles; the result relies on the use of linear programming, but, for a given number of degrees of freedom, it provides conditions which are both necessary and sufficient for stability.

The results of Chapter 3 provided conditions on the current input which guarantee

stability, but they did not, in themselves, lead to an algorithm which yields an optimum choice for that input. In Chapter 4, we considered an alternative procedure for dealing with short term infeasibility (STIF) by focusing on CSGPC and proposing modifications which maintain stability when, due to set-point changes, CSGPC encounters STIF. This was first implemented by simply minimizing the deviation of the predicted steady-state value of the output (the slack variable, s_∞) from its target value; while this approach has the advantage of simplicity, it ignores transient errors. For Mixed-Objective CSGPC, we shifted the objective to the minimization of the infinity-norm of the predicted errors. The final modification in Chapter 4, MCSGPC, was similar, but retained a two-norm cost and guaranteed an asymptotic return to short term feasibility (STF) by adding a slack variable end-point constraint. The modified algorithms are activated only when CSGPC is STIF, and guarantee recovery of STF; this, together with the properties of CSGPC, were shown to guarantee stability and asymptotic tracking.

In Chapter 5, we looked specifically at the necessity of previously proposed terminal constraints. While it had been recognised that, for the purposes of stability, one actually only needs to force the output errors to be stable, thereby turning the predicted output error trajectories into infinite length sequences (ILS), we showed that it was also the case that one need only force the input increments to be stable, with the effect of getting predicted input increment trajectories which are ILS; we proposed CaSC, CaML, and IHSPC which implement these changes in philosophy and derived two procedures for calculating the infinite horizon cost involving the sum of the square of the ILS errors and input increments. Then, through the use of suitable input horizon bounds, we developed simple, but efficient means of invoking input constraints over a infinite horizon by enforcing them over a finite horizon and subsequently implemented these constraint

horizons in CCaSC.

The final results of this thesis, presented in Chapter 6, dealt with the inherent clash between disturbances and constraints. We first considered how the effects of disturbances could be propagated forward in time and then derived necessary and sufficient limits on the size of the future inputs which are required to reject all possible (norm-bounded) future disturbances. We modified the constraint limits of CSGPC accordingly, so as to reserve the necessary control authority to reject these disturbance effects and to ensure that the problem would continue to be feasible at all subsequent time steps, and thus we developed an algorithm with guaranteed stability and asymptotic tracking for systems subject to disturbances.

7.2 Open problems

This thesis has dealt exclusively with single input, single output systems, but the results presented here can easily be extended to multivariable systems. Additionally, no results were presented on the effects of model uncertainty for the control of systems which are subject to physical constraints; guaranteeing stability under these conditions will prove to be a difficult task. Other areas, for which extension of these results has yet to be addressed, include constrained control of time varying systems, non-linear systems, and continuous time systems.

Bibliography

[1] J.C. Allwright (1993). "On mini-max model-based predictive control." in *Advances in Model Based Predictive Control,* D.W. Clarke (ed.), Oxford Science Publications.

[2] A. Bemporad and E. Mosca (1994). "Constraint fulfilment in feedback control via predictive reference management." *Proc. 3rd IEEE CCA*, Glasgow, 1909-1914.

[3] M. J. Best and K. Ritter (1985). *Linear programming: active set analysis and computer programs,* Prentice-Hall, 246.

[4] E.F. Camacho (1993). "Constrained generalized predictive control." *IEEE Trans. on Automatic Control*, **38**, 327-332.

[5] D.W. Clarke and P. Gawthrop (1975). "Self-tuning control." *Proc. IEE, Pt. D*, **122**, 929-934.

[6] D.W. Clarke and P. Gawthrop (1979). "Self-tuning controller." *Proc. IEE, Pt. D*, **126**, 633-640.

[7] D.W. Clarke, C. Mohtadi, and P.S. Tuffs (1987). "Generalized predictive control." Pts. 1 and 2. *Automatica*, **23**, 137-160.

[8] D.W. Clarke, and R. Scattolini (1991). "Constrained receding horizon predictive control." *Proc. IEE, Pt. D*, **138**, 347-354.

[9] C.R. Cutler and B.L. Ramaker (1980). "Dynamic matrix control - a computer control algorithm." *Proc. JACC*, San Francisco.

[10] H. Demircioglu and D.W. Clarke (1993). "Generalised predictive control with end-point state weighting." *Proc. IEE, Pt. D*, **140**, 275-282.

[11] R. Fletcher (1987). *Practical methods of optimization.* John Wiley.

[12] C.E. Garcia, D.M. Prett, and M. Morari (1989). "Model predictive control: theory and practice, a survey." *Automatica*, **25**, 335-348.

[13] E.G. Gilbert and K.T. Tan (1991). "Linear systems with state and control constraints: the theory and application of maximal output admissible sets." *IEEE Trans. on Automatic Control,* **36**, 1008-1020.

[14] E.G. Gilbert, I. Kolmanovsky, and K.T. Tan (1994). "Nonlinear control of discrete-time linear systems with state and control constraints: a reference governor with global convergence properties." *Proc. 33rd CDC*, Florida, 144-149.

[15] J.R. Gossner, B.Kouvaritakis and J.A. Rossiter (1995). "Cautious stable predictive control: a guaranteed stable predictive control algorithm with low input activity and good robustness." *Proc. 3rd IEEE Mediterranean symposium on new directions in control and automation*, Cyprus, **2**, 243-250, and to appear, *Intl. Journal on Control.*

[16] K.N. Hrissagis, O.D. Crisalle, and M. Sznaier (1995). "Robust predictive control design with guaranteed nominal performance." *Proc. ACC.*

[17] T. Jolly and J. Bentsman (1993). "Generalized predictive control algorithms with guaranteed frozen-time stability and bounded tracking error." *Proc. ACC*, California, 384-388.

[18] S.S. Keerthi and E.G. Gilbert (1988). "Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: stability and moving-horizon approximations." *Journal of Optimization Theory and Applications,* **57**, 265-293.

[19] B. Kouvaritakis, J.A. Rossiter, and A.O.T. Chang (1992). "Stable generalized predictive control." *Proc. IEE, Pt-D,* **139**, 349-362.

[20] B. Kouvaritakis, J.R. Gossner, and J.A. Rossiter (1996). "A priori stability conditions for an arbitrary number of unstable poles." *Proc. 13th World Congress, Intl. Federation of Automatic Control, IFAC '96*, San Francisco, and to appear, *Automatica.*

[21] W.H. Kwon and A.E. Pearson (1977). "A modified quadratic cost problem and feedback stabilization of a linear system." *IEEE Trans. on Automatic Control,* **22**, 838-842.

[22] C.L. Lawson and R.J. Hanson (1974). *Solving least squares problems*. Prentice Hall.

[23] D.Q. Mayne, and E. Polak (1993). "Optimization based design and control," *Proc. IFAC World Congress*, **3**, 129-138.

[24] H. Michalska and D.Q. Mayne (1993). "Robust receding horizon control of constrained nonlinear systems." *IEEE Trans. on Automatic Control,* **38**, 1623-1633.

[25] C. Mohtadi (1986). "Advanced self-tuning algorithms." D. Phil. Thesis, Oxford University.

[26] M. Morari (1994). "Model predictive control: Multivariable control technique choice of the 1990's," in *Advances in Model Based Predictive Control,* D.W. Clarke (ed.), 22-37. Oxford University Press, Oxford.

[27] E. Mosca and J. Zhang (1992). "Stable redesign of predictive control." *Automatica,* **28**, 1229-1233.

[28] G.D. Nicolao, L. Magni, and R. Scattolini (1994). "On robustness properties of constrained receding-horizon controllers." *Proc. 33rd CDC*, Florida, 3023-3024.

[29] D.M. Prett and R.D. Gillette (1979). "Optimization and multivariable control of a catalytic cracking unit." *Proc. JACC*, San Francisco.

[30] J.B. Rawlings and K.R. Muske (1993). "The stability of constrained receding horizon control." *IEEE Trans. on Automatic Control,* **38**, 1512-1516.

[31] J.A. Richalet, A. Rault, J.L. Testud, and J. Papon (1978). "Model predictive heuristic control: applications to a industrial process." *Automatica*, **14**, 413-428.

[32] B.D. Robinson and D.W. Clarke (1991). "Robustness effects of a prefilter in generalised predictive control." *Proc. IEE, Pt. D*, **138**, 2-8.

[33] J.A. Rossiter and B. Kouvaritakis (1993). "Constrained stable generalized predictive control." *Proc. IEE, Pt-D*, **140**, 243-254.

[34] J.A. Rossiter and B. Kouvaritakis (1994). "Robustness and efficiency of generalized predictive control algorithms with guaranteed stability." *Proc. IEE Conference, CONTROL 94*, Warwick, 1017-1022.

[35] J.A. Rossiter, B. Kouvaritakis, and J.R. Gossner (1994). "Constrained stable generalized predictive control: stability results and the case of disturbances." Oxford University Tech. Report, OUEL 2018/94.

[36] J.A. Rossiter (1994). "GPC controllers with guaranteed stability and mean-level control of unstable plant." *Proc. 33rd CDC*, Florida, 3579-3580.

[37] J.A. Rossiter, B. Kouvaritakis, and J.R. Gossner (1995). "Feasibility and stability results for constrained stable generalized predictive control." *Automatica,* **31**, 863-877.

[38] J.A. Rossiter, B. Kouvaritakis, and J.R. Gossner (1995). "Mixed objective constrained stable generalized predictive control." *IEE Proc. - Control Theory Applications*, **142**, 286-294.

[39] J.A. Rossiter, B. Kouvaritakis, and J.R. Gossner (1995) "Stable generalized predictive control in the presence of constraints and bounded disturbances." *Proc. 3rd European Control Conference, ECC '95*, Rome, 3241-3246.

[40] J.A. Rossiter, B. Kouvaritakis, and J.R. Gossner (1995). "Guaranteeing feasibility in constrained stable generalized predictive control." Oxford University Tech. Report, OUEL 2061/95, and to appear, *IEE Proc. - Control Theory Applications*.

[41] J.A. Rossiter, B. Kouvaritakis, and J.R. Gossner (1995). "Constrained cautious stable predictive control." Oxford University Tech. Report, OUEL 2066/95.

[42] J.A. Rossiter, B. Kouvaritakis, and J.R. Gossner (1996). "Infinite horizon generalized predictive control." To appear, *IEEE Trans. on Automatic Control*.

[43] P. Scokaert (1994). "Constrained predictive control." D.Phil. thesis, Department of Engineering Science, Oxford University.

[44] M. Sznaier and M.J. Damborg (1990). "Heuristically enhanced feedback control of constrained discrete-time linear systems." *Automatica,* **26**, 521-532.

[45] M. Sznaier and M.J. Damborg (1993). "Heuristically enhanced feedback control of constrained systems: The minimum time case." *Automatica,* **29**, 439-444.

[46] T.T.C. Tsang and D.W. Clarke (1988). "Generalized predictive control with constraints." *Proc. IEE, Pt-D*, **135**, 451-460.

[47] P. Vyas (1996). "Plasma vertical position control in the Compass-D tokamak." D.Phil. thesis, Department of Engineering Science, Oxford University.

[48] P. Vyas, J.R. Gossner, and B. Kouvaritakis (1996). "Application of cautious stable predictive control to vertical positioning in Compass-D tokamak." Oxford University Tech. Report, OUEL 2092/96.

[49] T-W. Yoon and D.W. Clarke (1995). "Observer design in receding horizon predictive control." *Intl. Journal on Control*, **61**, 171-191.

[50] P.C. Young, M.A. Behzadi, C.L. Wang, and A. Chotai (1987). "Direct digital and adaptive control by input output state variable feedback," *Intl. Journal on Control*, **46**, 1861-1881.

[51] E. Zafiriou (1990). "Robust model predictive control of processes with hard constraints." *Comp. Chem. Eng.*, **14**, 359-371.

[52] E. Zafiriou and H.-W. Chiou (1993). "Output constraint softening for SISO model predictive control." *Proc. ACC,* San Francisco, 372-376.

[53] Z.Q. Zheng and M. Morari (1993). "Robust stability of constrained model predictive control." *Proc. ACC,* San Francisco, 379-383.

[54] A. Zheng and M. Morari (1994). "Stability of model predictive control with soft constraints," *Proc. 33rd CDC*, Florida, 1018-1023.